A cuckoo search optimization-based forward consecutive mean excision model for threshold adaptation in cognitive radio

H. Abdullahi[1], A. J. Onumanyi[2,1,*], S. Zubair[1], A. M. Abu-Mahfouz[3,2], G. P. Hancke[4,2]

[1] Department of Telecommunication Engineering, Federal University of Technology, Minna, Nigeria
e-mail: mabdullahihassan@yahoo.com, zubairman@futminna.edu.ng
[2] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa
e-mail: adeiza1@futminna.edu.ng
[3] Council for Scientific and Industrial Research, Pretoria, South Africa, South Africa
e-mail: a.abumahfouz@ieee.org
[4] Department of Computer Science, City University of Hong Kong, Hong Kong, China
e-mail: gp.hancke@cityu.edu.hk

*Corresponding author

Abstract: The forward consecutive mean excision (FCME) algorithm is one of the most effective adaptive threshold estimation algorithms presently deployed for threshold adaptation in cognitive radio (CR) systems. However, its effectiveness is often limited by the manual parameter tuning process and by the lack of prior knowledge pertaining to the actual noise distribution considered during the parameter modelling process of the algorithm. In this paper, we propose a new model that can automatically and accurately tune the parameters of the FCME algorithm based on a novel integration with the cuckoo search optimization (CSO) algorithm. Our model uses the between-class variance function of the Otsu's algorithm as the objective function in the CSO algorithm in order to auto-tune the parameters of the FCME algorithm. We compared and selected the CSO algorithm based on its relatively better timing and accuracy performance compared to some other notable metaheuristics such as the particle swarm optimization (PSO), artificial bee colony (ABC), genetic algorithm (GA), and the differential evolution (DE) algorithms. Following close performance values, our findings suggest that both the DE and ABC algorithms can be adopted as favourable substitutes for the CSO algorithm in our model. Further simulation results show that our model achieves reasonably lower probability of false alarm and higher probability of detection as compared to the baseline FCME algorithm under different noise-only and signal-plus-noise conditions. In addition, we compared our model with some other known autonomous methods with results demonstrating improved performance. Thus, based on our new model, users are relieved from the cumbersome process involved in manually tuning the parameters of the FCME algorithm; instead, this can be done accurately and automatically for the user by our model. Essentially, our model presents a fully blind signal detection system for use in CR and a generic platform deployable to convert other parameterized adaptive threshold algorithms into fully autonomous algorithms.

Key words Adaptive Threshold – Autonomous – Cognitive Radio – FCME – Metaheuristic Algorithm – Parameter tuning

1 Introduction

Cognitive radio (CR) refers to an intelligent radio that automatically detects whether a channel is occupied or not and then adjusts its transceiver parameters in order to improve utilization and reliability of the communication channel [1, 2]. CR is currently being developed under the specifications of the IEEE 802.22 draft standard, which emphasizes that spectrum sensing (SS) remains a requirement for effective spectrum identification in CR systems [3–5]. Typically, SS involves routinely sensing the radio spectra to determine whether primary user (PU) signals are present or not in a specified channel. SS can be realized using different detectors/techniques such as the energy detector (ED), the cyclostationary detection (CD) technique, the Eigen-value, the covariance, the prediction-based methods [6], and the evolutionary-based methods [7–10]. However, the ED is widely considered because it is simple, fast, and does not need to know the waveform/type of the incumbent PU signal [11, 12].

To sense, the ED simply measures the received signal energy in a channel and estimates a threshold value,

which it uses to decide whether PU signals are present or not in a given channel [13]. The ED declares a channel as occupied if the received signal energy exceeds the estimated threshold value and declares the channel as unoccupied (free/whitespace) if the received signal energy is equal to or below the threshold value. Theoretically, the ED estimates its threshold value using some prior knowledge of its average noise level. However, this static approach introduces severe decision errors because noise levels are typically random and are largely affected by unknown ambient conditions. Furthermore, this randomness makes it difficult to estimate accurate threshold values particularly under dynamic spectra conditions [14, 15].

Consequently, most modern EDs resort to the use of adaptive thresholds in order to react quickly to fluctuating channel conditions [16, 17]. Thus, adapting the ED's threshold value has led to the design of different adaptive threshold estimation techniques (ATT) in the literature with the forward consecutive mean excision (FCME) algorithm being one of the most attractive options [18]. The FCME algorithm is well known for its blind sensing capability, simple design, and computational efficiency [18]. However, the FCME algorithm is only as effective as its highly sensitive parameter values namely, the threshold factor, T_{cme} and the percentage of the initial clean sample set, Q. In most related works, T_{cme} is usually computed based on some pre-known or assumed noise distribution in order to maintain a target probability of false alarm, P_{FA} for the FCME algorithm [16, 17]. In some other cases, T_{cme} is determined based on some pre-determined noise-only sample set [19, pp.1129] [20, pp.2]. However, these approaches to computing T_{cme} as well as Q are not always accurate because radio spectra conditions are highly dynamic, which makes it difficult to establish global parameter values. Other problems associated with setting the parameters of the FCME algorithm include the inability to recompute these parameters manually during online operations and the possibility of estimating wrong parameter values due to faulty assumptions about the noise distribution under consideration. These problems may plague the FCME algorithm into computing either too high or too low threshold values causing the ED to render wrong occupancy decisions. These wrong decisions may thus cause severe interference to PUs or gross spectra underutilization by the CR [18].

Thus, in this paper, we have developed a new model that automatically and accurately determines the optimal parameters of the FCME algorithm based on the measured dataset considered per time. Interestingly, following our new model, users of the FCME algorithm no longer need to compute or tune manually the parameters of the FCME algorithm, instead our model achieves this automatically and accurately for the user. In addition, our model still supports a manual configuration module, particularly for users who may yet desire to enforce a specific target P_{FA} rate for the algorithm. Essentially, our model allows the FCME algorithm to be used either as an *adaptive-only* algorithm (when manual parameter tuning is used) or as a fully self-configurable algorithm (when auto-tuning is used). Our model uses the cuckoo search optimization (CSO) algorithm to auto-tune the parameters of the FCME algorithm. In the CSO algorithm, our model adopts the between-class variance function of the Otsu's algorithm as the objective function to determine the optimal parameter values of the FCME algorithm. Furthermore, in order to keep the FCME algorithm fully autonomous, we have developed a new heuristic that accurately estimates threshold values particularly in noise-only conditions without requiring the need for a predefined target P_{FA} rate by the user. These ideas have not only improved the FCME algorithm's performance concerning SS in CR, but they have as well successfully extended the FCME algorithm into an entirely self-configurable threshold estimation algorithm. Interestingly, our model can be easily extended to other highly parameterized threshold estimation algorithms in the literature to make them fully autonomous.

The rest of the paper is structured as follows: Section 2 provides a brief literature review with regards to the FCME algorithm. In Section 3, we provide the system under consideration and the background algorithms required for our design. A full exposition of our model is provided in Section 4. The method of analysis used in our work is described in Section 5. Results are presented and discussed in Section 6, while conclusions are drawn in Section 7.

2 Related work

In this section, we discuss related works in order to motivate two major concerns. First, to establish the fact that till date, there exist no fixed global parameter values for the FCME algorithm. This lack implies that there is need to develop standard and automatic methods to optimize the FCME algorithm's parameters. Second, we highlight the main approaches used in the literature to compute the FCME algorithm's parameters. We discuss different versions of the FCME algorithm in order to motivate the uniqueness of our approach (or model) as proposed in this paper.

From our analysis of the related works presented in Table 1, we found that no global parameter value exists that guarantees the highest performance of the FCME algorithm in all possible conditions. An obvious reason is that the radio spectrum is highly dynamic, which makes it impractical to adopt a global value. This implies that authors may not always adopt the best method in order to configure the FCME algorithm's parameters, which will ultimately undermine the algorithm's performance when spectra conditions suddenly change. Thus, it becomes quickly obvious that standard methods with the

Table 2.1: Summary of Review of Related Literature

S/N	Year/Reference	$\mathbf{T}_{cme}$	$\mathbf{Q}(\%)$	$\mathbf{P}_{FA}(\%)$
1	2004 [16]	2.97	10	-
2	2004 [21]	2.97	10	0.1
3	2004 [22]	2.97	10	-
4	2005 [23]	-	10	-
5	2005 [20]	2.97	10	0.1
6	2005 [24]	2.8703	25	-
7	2006 [25]	6.91	10	0.1
8	2007 [17]	1.4	25	0.01
9	2007 [26]	2.53	25	0.01
10	2008 [27]	6.9078	20	0.1
11	2009 [28]	-	6.25	0.1
12	2010 [29]	2.3, 4.6, 6.9	-	-
13	2011 [30]	13.81, 4.61	-	10^{-6}, 0.1
14	2011 [18]	4.6052	10	-
15	2012 [31]	4.6052	-	0.01
16	2012 [32]	Multiple values	-	-
17	2015 [33]	2.99	-	0.05
18	2016 [34]	-	-	0.01
19	2016 [35]	-	10	0.01

capacity for automation are required. Such methods are clearly unavailable in the literature. For example, concerning estimating the Q parameter of the FCME algorithm, so many authors prefer to use $Q = 10\%$ as the *de factor* value because it was found to be optimal after several experiments were conducted. However, we found (see Table 2.1) that some authors used higher values such as $Q = 25\%$ in [24], while lower values were also used ($Q = 6.25\%$) by the same authors in [28]. The T_{cme} parameter has also been computed based on prior assumptions concerning the underlying noise distributions. In many such works (see [16, 17]), Gaussian distributions were assumed to describe the underlying noise process. In some other cases, the exponential distribution was assumed and used to model the noise process [32]. Unfortunately, if such assumptions turn out to be incorrect, then the performance of the FCME algorithm becomes highly undermined leading to severe implications such as increased interference to PUs or gross spectra underutilization by the CR. We found it highly contemporary and compelling to develop new models that will not only standardize the configuration process, but also automate the process as well.

In addition to auto-tuning the FCME algorithm's parameters, some other authors have tried to improve the accuracy of the algorithm. For example, authors in [36] proposed the median filtered FCME (MED-FCME) to enhance the baseline FCME algorithm (that is, the original FCME algorithm [19]). The MED-FCME algorithm performed better than the baseline FCME algorithm in bands with high noise variability and very high signal occupancy rates [36]. Unlike the baseline FCME algorithm, the MED-FCME algorithm pre-filters the input signal using a median filter of a defined length. This pre-filtering process reduces the noise variance making it easier to estimate accurate threshold values. The MED-FCME algorithm also protects against improper estimates of the T_{cme} parameter of the baseline FCME algorithm based on its pre-filtering action.

The localization algorithm based on double thresholding (LAD) with adjacent cluster combining (ACC), termed LAD ACC was also developed based on the baseline FCME algorithm. The LAD ACC FCME algorithm addresses issues related to signal localization. The LAD ACC estimates channel occupancy and duty cycle rates accurately by reducing the probability of miss-detection [31]. However, the LAD ACC does not modify nor enhance the baseline FCME algorithm; instead it simply uses the baseline FCME algorithm to estimate double threshold values to improve the probability of signal detection in CR. The $m - dB$ approach is another variant of the baseline FCME algorithm [31, 37]. The $m - dB$ model simply adds a tolerance value, m, to the threshold value computed by the FCME algorithm. This added tolerance value maintains a target P_{FA} rate for the baseline FCME algorithm.

However, in this paper, we propose a new model that differs from the MED-FCME, LAD ACC, and the $m - dB$ models. Our model standardizes and automates the parameter tuning process of the baseline FCME algorithm. Unlike other models that keep the parameters of the baseline FCME algorithm fixed, instead, our model accurately auto-tunes the algorithm's parameters based on the particular input dataset under consideration per time. Apart from providing a new functionality to the FCME algorithm, our model can be extended to other parameterized threshold estimation algorithm to make them fully autonomous. This benefit motivated the ideas proposed in this paper.

To conclude, since our model concerns adopting metaheuristic algorithms to fine-tune the paramters of the FCME algorithm, it is worthwhile to mention here a few methods used for meta-optimization. Meta-optimization describes the use of an optimization method to tune another optimization method. A few notable methods among others for this purpose include the Tuning with chess rating system (CRS-Tuning) [38], the racing algorithm based on Friedman test (F-Race) [39], and the method for relevance estimation and value calibration (REVAC) [40]. These methods are considered essential when it concerns the need to fairly compare different metaheuristic algorithms [41]. Other attempts have been considered such as determining the critical values for the control parameters of Differential Evolution algorithms [42], and the use of Genetic Algorithm in addressing some security issues in CR considering cooperative spectrum sensing [43–45]. These meta-optimization methods mentioned above can be used to improve the per-

formance of metaheuristic algorithms particularly under the deeper context of parameter-tuning in adaptive threshold algorithms. Although the use of meta-optimization is at the moment outside the scope of our model, nevertheless, we consider it worthy of mention to-wards the advancement of adaptive threshold estimators in CR.

3 System Description and Background Algorithms

In this section, we describe the energy detector (ED) in which our proposed model is deployed. Then we describe the FCME and the CSO algorithms.

3.1 ***The Energy Detector***

The ED is modeled as shown in Fig. 3.1. Here, we describe the flow process of the ED used in our research. Each simulated signal set considered in our test experiments (in the result section) were obtained using the ED as follows: The transmitted signal samples were influenced by the transmission channel and corrupted by noise. We modeled the received signal, $y(t)$ as

$$y(t) = h(t)x(t) + w(t) \tag{3.1}$$

where t is the time index, $h(t)$ is the channel impulse response function, $x(t)$ is the PU transmitted signal, and $w(t)$ is the system noise described as additive white Gaussian noise (AWGN). For simulation purpose, we modeled the channel as Rayleigh fading noting that most wireless communication channels in urban environments are characterized by the Rayleigh distribution, including CR networks.

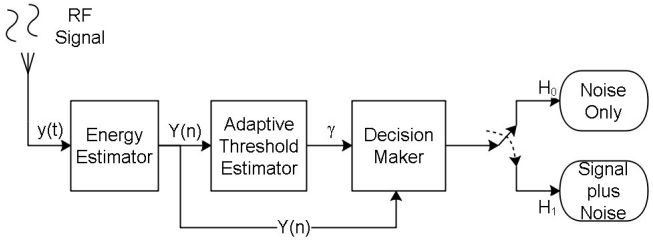


Fig. 3.1: The energy detection system

Essentially, the ED computes the power spectral density (PSD) of $y(t)$ as $Y\ (n)$, where n denotes the frequency channel index. To compute the PSD, the fast Fourier transformation (FFT) algorithm was used. The PSD samples, $Y(n)$, were considered as the test statistic in our experiments. Following Fig. 3.1, the PSD samples were then passed to the adaptive threshold estimator block in order to estimate an appropriate thresh-old value, γ. The test statistic, $Y(n)$, was then compared to γ to determine the state of the channel in the decision maker block. For any channel to be declared vacant (noise-only samples), H_0, it was required that $Y(n) \leq \gamma$, and to be declared occupied (signal-plus-noise samples), H_1, requires that $Y(n) > \gamma$. These hypotheses, H_0 and H_1 are generally defined considering the frequency domain samples as:

$$H_0 : Y(n) = W(n), \text{ for } n = 1, 2, ..., N \tag{3.2}$$

$$H_1 : Y(n) = H(n)X(n) + W(n), \text{ for } n = 1, 2, .., N \tag{3.3}$$

where N is the total number of frequency channels and other variables are obtained after applying the FFT to Eq. (3.1). In this case, we computed N as

$$N = 2^{\lceil (\log_2(T * f_s) - 1) \rceil} \tag{3.4}$$

where T is the total sensing period and f_s is the sampling frequency. Summarily, our goal in the entire detection process is to determine either H_0 or H_1 using an accurately estimated γ. Consequently, in subsequent sections, we will concern ourselves with the approach used to estimate γ.

3.2 ***FCME algorithm***

The FCME algorithm is described in this section. The FCME algorithm is an efficient, effective and simple algorithm, no wonder its been used extensively in CR [46]. The FCME algorithm consists of the following necessary parameters:

1. T_{cme} is called the threshold factor, which is used to control the threshold value γ estimated by the FCME algorithm. It is computed based on a P_{FA} (probability of false alarm) value usually predefined by the user in a typical formula such as $T_{cme} = -\ln(P_{FA})$[25]. The FCME algorithm then computes γ by multiplying T_{cme} by the estimated mean noise level in the specified band. Estimating accurate T_{cme} values is important because using high T_{cme} values yields a high threshold value that may cause high miss detection rates (interference to PUs). If the estimated T_{cme} value is too small, then this may yield low threshold values causing high false alarm rates (spectra under-utilization by the CR).
2. Q denotes the percentage of the initial clean sample set used in the FCME algorithm. This parameter determines the number of noise-only samples considered by the FCME algorithm in order to compute γ. It is widely defined as $Q = 0.1 * N$, that is, 10% of the total number of samples [25]. If large Q values are used, the FCME algorithm may use more signal samples thus corrupting the set and leading to large γ values causing high miss detection rates. On the other hand, using small Q values may lead to long processing/computational time and divergence of the algorithm. Consequently, it is important to choose/ estimate the parameter values of the FCME

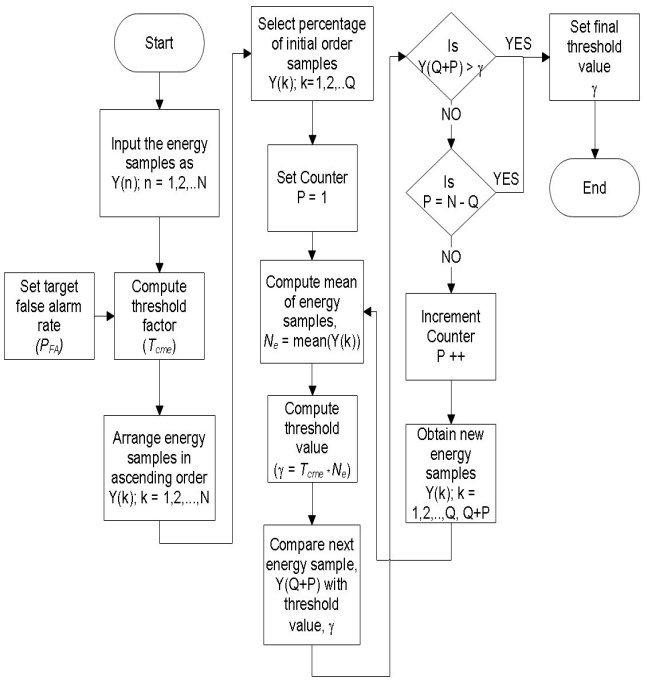


Fig. 3.2: The flow process of the FCME algorithm

algorithm carefully. Having understood the parameters of the FCME algorithm, we now present how the FCME algorithm works.

Fig. 3.2 depicts the flow process of the FCME algorithm [25]. According to [25], the algorithm accepts the estimated energy samples, $Y(n)$ $n = 1, 2, ..., N$, as input. In the baseline FCME algorithm, the T_{cme} and Q parameters are calculated *a priori* (manually) at the start of the process. Usually, since these parameters are manually computed, they remain unchanged until either the system becomes idle again or the user shuts down the algorithm in order to reconfigure its parameters. Once the algorithm's parameters are defined, then $Y(n)$, $n = 1, 2, ..., N$ is rearranged in an ascending order as $Y(k)$, $k = 1, 2, ..., N$. The algorithm begins with the first Q samples in $Y(k)$, i.e. $Y(k)$, for $k = 1, 2, ..., Q$. The mean of the first Q samples in $Y(k)$ is obtained as $N_e = \frac{1}{Q}\sum_{k=1}^{Q} Y(k)$. Then an initial threshold value is computed as $\gamma = T_{cme} \times N_e$. The next sample, $Y(Q+P)$ is compared with γ, where P is a counter initially defined as $P = 1$. If $Y(Q+P)$ is greater than γ, then $Y(Q+P)$ is considered as an outlier (signal sample) and the FCME algorithm terminates returning γ as the final threshold value. However, if $Y(Q + P)$ is less than γ, the algorithm updates its counter ($P = P + 1$) and re-iterates until an outlier is found or a limiting value is reached (i.e. if $P = N - Q$), which terminates the algorithm and converges to a final γ value.

Algorithm 1: The CSO Algorithm

Data:

1. Number of variables (dimensionality), D
2. Constraints: Lower and upper bounds of each variable
3. Objective function, F

Control parameters:

1. Number of nests (population size), I
2. Number of Iterations, Z
3. Probability of discovering alien egg, P_a
4. Total number of fitness function consumed, ToT_Eval

Result:

1. Best Fitness value, $F_{\max}$
2. Optimal variables, x_i

Process:

```
// Initialization phase:
numeval ← 0
for i = 1 to I do
    x_i ← rand()
    evaluate(x_i) based on Eq. 4.1 (i.e. the objective function)
    numeval++
end
// Obtain current best solution as follows:
F_max ← max(entiresolution)
// Main loop begins here:
while z < Z do
    // Carry out Lèvy flight phase here:
    for i = 1 to I do
        Get a new solution randomly by Lèvy flight using Mantegna's algorithm
        evaluate(x_i) based on Eq. 4.1
        numeval++
        // Obtain the new global best value:
        F_max ← max(entiresolution)
        if F_i > F_max then
            // Update the new global best
            F_max = F_i
        else
            // Keep current global best value:
            F_max = F_max
        end
        Memorize best value and terminate process if numeval ≥ ToT_Eval
    end
    // Abandon nest phase
    for i = 1 to I do
        Empty a fraction of the worst nest based on P_a using biased/selective random walks
        Update the new solution using Eq. 3.5
        evaluate(x_i) based on Eq. 4.1
        numeval++
        // Keep the new best solution
        if F_i > F_max then
            // Update the new global best
            F_max = F_i
        else
            // Keep current global best value
            F_max = F_max
        end
    end
    Memorize best value and terminate process if numeval ≥ ToT_Eval
end
Post process and visualize the results
```

3.3 ***Cuckoo search optimization algorithm***

We acknowledge that many optimization algorithms exist in the literature and we do not claim to have evaluated exhaustively in our work all existing metaheuristics. Nevertheless, we have compared in this paper the CSO algorithm with some few other notable metaheuristics. Our findings suggest that the CSO algorithm presents convincing performance in terms of its convergence to the best fitness value under different CR working conditions (kindly see the result section in this regard). Furthermore, since the CSO algorithm depends on a minimal set of parameters to be fine-tuned, we have considered the tuning process as simple enough for use in our model. Thus, convinced by its performance, we briefly describe its general workings as follows (readers are kindly referred to [47] for further details concerning the CSO):

The CSO algorithm based on *Lèvy* flight works using the following model [47]:

$$x_i^{(z+1)} = x_i^{(z)} + \alpha \otimes L\grave{e}vy(\lambda) \tag{3.5}$$

where $x_i^{(z+1)}$ denotes a new solution for a cuckoo, i, using *Lèvy* flight function per iteration, z, with λ being the *Lèvy* walk parameter, α is the step size related to the scale of the problem of interest, and $\otimes$ symbol means entry wise multiplication. The *Lèvy* flight provides a random walk, while the random step length is drawn from a *Lèvy* distribution $L\grave{e}vy \sim u = v^{-\lambda}$, $(1 < \lambda \leq 3)$. According to [47], the CSO algorithm mimics the biological hatching process adopted by Cuckoo birds. Usually, a Cuckoo bird's egg is laid in a foreign bird's nest so that it may be hatched by the foreign bird. In the technical sense, each cuckoo egg represents a new solution within a nest containing other eggs (i.e the population of possible solutions). The aim is to use the new and potentially better solutions (cuckoo eggs) to replace the not-so-good solutions (other eggs) in the nest. Following Algorithm 1, the CSO is popution-based with a population size (I). Each solution (egg) in the population (nest) is denoted as x_i. The number of times the fitness function is evaluated is denoted as *numeval*. To begin, an initial population is randomly generated in the *Initialization phase* (see Algorithm 1) wherein each individual solution x_i represents the real-valued vector with D elements (the dimensionality of the problem). The algorithm then iterates Z times per individual solution in the population towards determining the maximum fitness value (corresponding to the best solution). We seek the maximum value because our case concerns a maximization problem. Through the CSO process, each individual is improve towards better values based on Eq. 3.5. The process involves adopting the *Lèvy* flight function based on Mantegna's algorithm (kindly see [47] for details). Thereafter, the algorithm empties poorer solutions from the nest and replaces them with better solutions during the *Abandon nest phase*. The algorithm continues to iterate until a stopping condition is reached, which is considered to be the maximum number of fitness function consumed (Kindly see the Result Section).

4 Proposed CSO-FCME model

In this section, we describe our proposed CSO-FCME model and then we present the overall summary of the algorithmic flow of the CSO-FCME model. We describe the choice/justification of the objective function used in the CSO-FCME model followed by a detailed description of each sub-algorithm that operates the CSO-FCME model.

4.1 ***Overview of the CSO-FCME model***

Fig. 4.1 presents our proposed CSO-FCME model and it is briefly described. Our model provides two basic functions for users of the FCME algorithm. It provides a module, $M4$, which enables a user to configure manually the parameters of the FCME algorithm in order to enforce a target P_{FA}. Module $M5$ provides the other functionality, which engages the self-configurability (auto-tuning capability) of our proposed model. The model obtains the input signal via module $M1$ and feeds the input signal to the FCME algorithm in module $M2$ in order to compute threshold values and feeds to the output module $M6$ in order to decide concerning the status of each channel.

Module $M3$ uses switch $SW2$ to engage either modules $M4$ or $M5$. If $M4$ is selected, then the user manually assigns the required parameter values of the FCME algorithm as $\{T_{cme}, Q\}$. The user will normally compute T_{cme} based on a predefined target P_{FA}, decided by the user. Both fixed values of T_{cme} and Q are fed via $SW2$ to $M2$ where a final threshold value (FTV), γ is computed by the FCME algorithm using $\{T_{cme}, Q\}$ and passed through switch $SW1$ to module $M6$. Then, $M6$ uses γ and the input signal set, $Y(n)$, for $n = 1, 2, ..., N$ to decide whether the channels are occupied or not.

On the other hand, if the user selects $M5$ using $SW2$, then our model engages its auto-tuning capability. In this case, random parameter values (PVs) are generated by the CSO algorithm running in $M5$ and these PVs are sent to $M2$ via $SW2$. The FCME algorithm in $M2$ uses the PVs to estimate a corresponding set of estimated threshold values (ETVs). These ETVs are then fed back to $M5$ via $SW1$ to be evaluated by the CSO algorithm using the proposed objective function (discussed in Section 4.3). Based on the fittest threshold value in the set of ETVs, the CSO algorithm (in $M5$) iterates in order to determine the optimal final parameter values (PVs), termed $\left\{T_{cme}^{best}, Q^{best}\right\}$. Module $M5$ returns $\left\{T_{cme}^{best}, Q^{best}\right\}$ to $M2$ via $SW2$ to compute FTV, γ^{best}

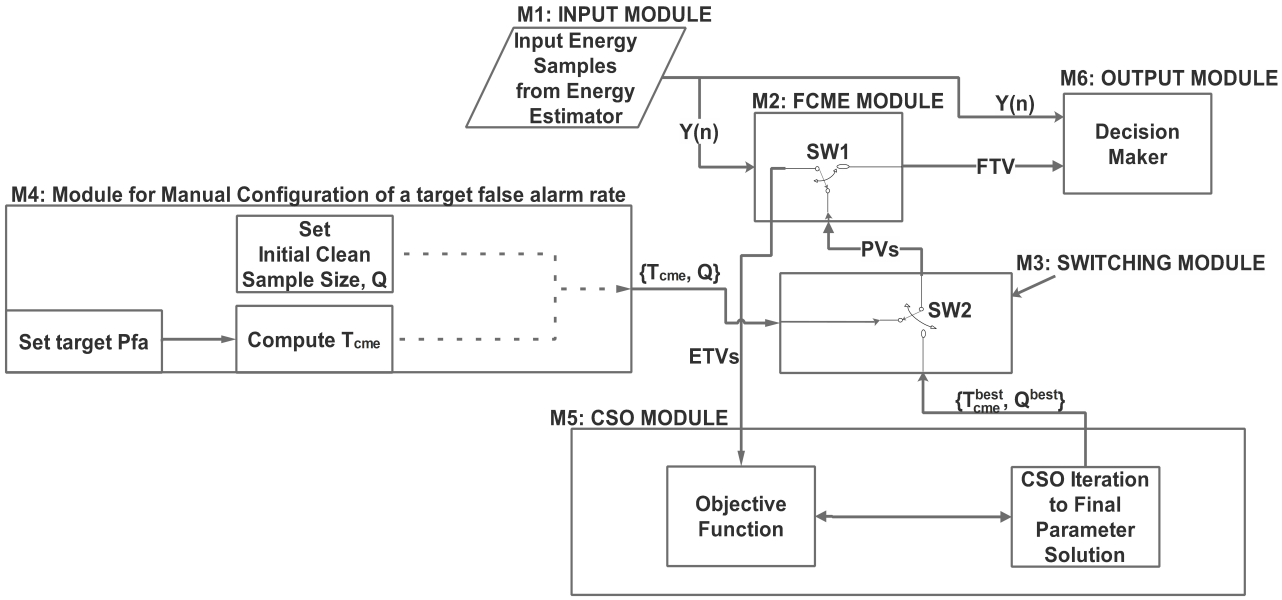


Fig. 4.1: Our proposed self-reconfigurable CSO-FCME model

(we used γ^{best} for the auto-tuning case). Then, γ^{best} is fed to $M6$ through $SW1$ in order to decide on the status of each channel.

Essentially, we have described two approaches by which our model enables the FCME algorithm to compute threshold values. One approach is to use a manual process (baseline FCME algorithm) and the other approach involves using the auto-tuning process (CSO-FCME model). In the next subsection, we describe the algorithmic process that governs the CSO-FCME model.

4.2 ***Algorithmic process of the CSO-FCME model***

Algorithm 2 presents a summary of the overall process involved in the CSO-FCME model. In this case, we describe only the auto-tuning component of our model, since the manual function is straightforward as in the baseline FCME algorithm. We relate the different algorithms that animate the modules, which have been described in Section 4.1. Kindly note that each algorithm mentioned here will be fully described in the next subsections.

Typically, our model runs Z different iterations based on the CSO algorithm (recall Algorithm 1). In each iteration, the CSO algorithm generates an initial set of parameter values (PVs) using the random initial solution generator (RISG) algorithm, which we shall discuss in the next subsections. Our model then evaluates these PVs using the fitness evaluation (FE) algorithm to determine the global best solution in the current iteration (still in module $M5$). The process then generates a new set of parameter values (new solutions) using the new solution generator (NSG) algorithm for the next iteration (in module $M5$). After each iteration, the CSO-FCME model continues to update the global best solution until no better solution is obtained or after Z iterations are completed. Then it obtains its final parameter values to compute the final optimal threshold value (in module $M2$). Thereafter, our model runs the noise only detector (NOD) algorithm to determine whether the input signal contains only noise samples (in module $M5$). Once this is confirmed, the algorithm outputs the final optimal threshold value, γ^{best} (in module $M2$). Module $M2$ passes γ^{best} to module $M6$ to decide on the channel's status based on the input signal from module $M1$. Having highlighted the respective algorithms involved in the CSO-FCME model, we shall now describe the objective function used and provide details of the individual algorithms in the following subsections.

4.3 ***Objective function used in the CSO algorithm***

Our model sets up an optimization problem in order to auto-tune the parameters of the FCME algorithm. Similar to every optimization problem, an objective (or fitness) function is required to evaluate the performance of each solution towards determining the best (optimal) solution. In our model, we used the between-class variance (BCV) function (see Eq 4.1) of the Otsu's algorithm [48] as our objective function. This function is used widely in image processing to evaluate the *goodness* of a threshold value used for image segmentation. However, different from its use in image processing, we have used the BCV function in our model to classify radio power samples into two classes namely, either the noise or the signal-plus-noise class. We used the BCV function since

Algorithm 2: Algorithmic process of the CSO-FCME model

Data:

1. PSD, $Y(n)$ $n = 1, 2, ..., N$; Lower and upper constraints, Lb and Ub

Result:

1. Final set of optimized parameter values (PV), $\Omega^{best} = [T_{cme}^{best}, Q^{best}]$,
2. The optimized final threshold value (FTV), γ^{best}

Process:

for *i = 1 to Z* **do**
  Run the CSO algorithm described in Algorithm 1 based on the initial values obtained using the **RISG** algorithm
  Evaluate the new solutions using the **FE** algorithm
  Obtain new solutions using the **NSG** algorithm
end
The **NOD** algorithm is used at this point in order to handle the case for the noise-only dataset

it allows us to test an estimated power threshold value directly and accurately using only the first order statistical parameters of the dataset under consideration per time. This characteristic of the BCV function makes it highly suitable for use in our model. We obtained clues about how viable the BCV function may be based on the work done in [13]. Authors in [13] showed that a modified Otsu's algorithm effectively computes threshold values based on the BCV function. However, different from the idea in [13], in this paper, we have used the BCV function in an innovative manner to find the optimal parameter values of the FCME algorithm. This idea forms a novel contribution towards the enhancement of the FCME algorithm. The BCV function is given as

$$F(\gamma) = p_s(\gamma) \times p_n(\gamma) \times [\mu_s(\gamma) - \mu_n(\gamma)]^2, \qquad (4.1)$$

where, $F(\gamma)$ is the BCV (i.e. the degree of separation) computed as a function of the threshold value γ, which is typically estimated by the FCME algorithm. p_s is the probability of the class of estimated signal samples as a function of the threshold γ, p_n is the probability of the class of estimated noise samples, μ_s is the mean of the class of estimated signal samples, and μ_n is the mean of the class of estimated noise samples, all computed as a function of the threshold value estimated by the FCME algorithm. Readers are kindly referred to [48] for further details concerning the BCV function. In our model, we used the BCV function to compute the *fitness* (degree of separation) of each threshold value in the ETV set in order to determine the optimal parameters of the FCME algorithm. The optimal threshold value computed using the optimal parameter values of the FCME algorithm is obtained as:

$$F(\gamma^*) = \max_{\gamma} F(\gamma) \qquad (4.2)$$

where γ^* is the optimal threshold value determined over the range of γ (ETVs), that is, over the range from $\min Y(n)$ to $\max Y(n)$ for $n = 1, 2, ..., N$.

Thus, we maximize Eq. (4.2) subject to the following parameter constraints:

1. $T_{cme}^{(L)} \leq T_{cme} \leq T_{cme}^{(H)}$, where T_{cme} is the threshold factor to be optimized, $T_{cme}^{(L)}$ and $T_{cme}^{(H)}$ are the lower and upper values of the T_{cme} factor, respectively.
2. $Q^{(L)} \leq Q \leq Q^{(H)}$, where Q is the percentage of the initial clean sample set (ICSS), $Q^{(L)}$ and $Q^{(H)}$ are the lower and upper values of Q.

In subsequent subsections, we concisely represent the above constraints as $Lb = [T_{cme}^{(L)}, Q^{(L)}]$ for the lower set of parameter constraints and $Ub = [T_{cme}^{(H)}, Q^{(H)}]$ for the upper set of parameter constraints. Also, though the lower and upper constraints can assume any values configured by the user, however, we used the following bounds in our experiments: $0.1 \leq T_{cme} \leq 5$ and $1 \leq Q \leq 25$. We selected these values based on the minimum and maximum values that have been used by authors in the literature (see Table 2.1).

4.4 ***Description of the algorithms used in the CSO-FCME model***

4.4.1 ***Data Offset Process*** The baseline FCME algorithm works best when the input data is positive valued because it usually multiplies the T_{cme} by the noise mean (recall Fig. 3.2). Notice that if the noise mean is negative valued, the FCME algorithm produces a smaller threshold value (more negative-valued threshold) because of the multiplicative effect, which undermines its performance. However, in most cases, PSD datasets can as well be negative valued especially when measured in decibel-milliwatts (dBm) units. In this case, the baseline FCME algorithm would typically convert the dBm values to watt before it computes a threshold value (see [16, 17]). However, this is an expensive process that requires multiplicative ($O(N^2)$) and logarithmic ($O(G(N) \log N)$) operations, where $O(\cdot)$ denotes the time complexity, $G(N)$ is the complexity of the chosen multiplication algorithm, and N is the variable sample length. Instead of this expensive process, we propose a simple offset function that requires only additive ($O(N)$) and subtractive ($O(N)$) operations as follows:

$$S(n) = Y(n) - Y_{\min}, \quad for\ n = 1, 2, ..., N \qquad (4.3)$$

where $Y_{\min} = \min(Y(n))$, for $n = 1, 2, ..., N$ denotes the smallest value in the dataset. Our model then processes the positive-valued dataset, $S(n)$ instead of the

negative-valued input dataset, $Y(n)$. However, the original negative-valued dataset can be easily retrieved when the process ends as follows:

$$Y(n) = S(n) + Y_{\min}, \quad for\ n = 1, 2, ..., N. \qquad (4.4)$$

One main advantage of the offset function is that it allows the FCME algorithm to be used on either positive or negative valued datasets. Similarly, our model converts the final estimated positive threshold value to its corresponding negative value using Eq. (4.4) by adding $Y_{\min}$ to the estimated positive-valued threshold. Consequently, following the offset function, we shall consider $S(n)$ instead $Y(n)$ in the processes of subsequent algorithms.

4.4.2 ***Random Initial Solution Generator*** The next task in our CSO-FCME model is to obtain the initial parameter values (PVs) or initial solutions. To achieve this, we propose the Random Initial Solution Generator (RISG) adapted to work in the CSO algorithm. The RISG generates a random set of PVs (recall Fig. 4.1) that begins the optimization process. The RISG takes the predefined constraints, Lb and Ub as inputs while it produces the PVs, denoted as $\Omega = [T_{cme}, Q]$, and the initial set of fitness values F produced by each set of parameters $[T_{cme}, Q]$ in Ω. Typically, Ω is a matrix of size $(I \times D)$, where I is the total number of PVs (or population of initial solutions) and D is the number of parameters, (i.e. recall that $D = 2$). Thus, each row in Ω contains two values, where the value in the first column is a randomly generated T_{cme} value and the value in second column is a randomly generated Q value. Then, the RISG iterates from $i = 1$ to I and through a nested loop from $d = 1$ to D in order to generate the complete set of ETVs, $\Omega^{(I\times D)}$. For each iteration i and d, the RISG applies a Uniform random function, $rand(.)$, based on the constraints, Lb and Ub to obtain $\Omega^{(i\times d)}$, which denotes a single row value in $\Omega^{(I\times D)}$. The RISG repeats this process for every iteration, z of the CSO-FCME model, i.e. from $z = 1, 2, ..., Z$ (recall Algorithm 1). Thus, when the RISG completes its process, it outputs $\Omega_z^{(I\times D)}$ and $F_z^i = 0$, for $i = 1, 2, ..., I$ for a single iteration, z. The PVs in $\Omega_z^{(I\times D)}$ will then be evaluated by the fitness evaluator (FE) to be described next. However, a summary presentation of the RISG is provided in Algorithm 3.

Algorithm 3: Random Initial Solution Generator (RISG)

Data:
1. The lower limits, Lb, and upper limits, Ub.

Result:
1. $\Omega_z^{(i\times D)}$, which denotes the individual PVs to be evaluated in the current iteration, z, of the RISG. Here, i denotes the index of each pair of parameters (or each nest in the population of initial solutions). The final set of all PVs are collated in $\Omega_z^{(I\times D)}$, where I denotes the total population of nests (initial solutions), and D is the number of parameters to be optimized. Here, $D = 2$. Essentially, the first column in $\Omega_z^{(I\times D)}$ contains the initial sets of T_{cme} values while the second column contains the initial sets of Q values.
2. The initial set of evaluated fitness values, F_z^i for $i = 1, 2, ..., I$, corresponding to each parameter in $\Omega_z^{(I\times D)}$.

Process:
1 **for** *i = 1 to I* **do**
2 **for** *d = 1 to D* **do**
3 $\Omega_z^{(i,d)} = Lb + (Ub - Lb) \otimes rand(D)$, where D is the length of the uniform random number generated between 0 and 1. The function *rand* represents the uniform random number function.
4 **end**
5 **end**

4.4.3 ***Fitness Evaluation (FE) algorithm*** The FE algorithm evaluates each parameter set in $\Omega_z^{(I\times D)}$ using the objective function in Eq. (4.1). To achieve this, first, the FE algorithm uses each set of parameter values (i.e. each row) in $\Omega_z^{(IxD)}$ to compute a corresponding threshold value, γ_{z+1}^i using the FCME algorithm. Second, the FE algorithm then evaluates each estimated threshold value, γ_{z+1}^i using Eq. (4.1) to obtain a new set of fitness values, F_{z+1}^i for $i = 1, 2, ..., I$. We present a summary of the FE algorithm below in which we provide details concerning how the parameters of the objective function in Eq. (4.1) are computed in steps 6 to 7. The FE algorithm is detailed in Algorithm 4.

4.4.4 ***Determination/Update of new solutions*** The CSO-FCME model estimates new solutions using the new solution generator (NSG). The NSG takes the output of the FE algorithm as its input and obtains the global best fitness value, $F^{\max}$ corresponding to the global best parameter values, Ω^{best} (i.e. the PVs in Fig. 4.1) and the global best threshold value, γ^{best} (recall Section 4.1). Essentially, the NSG compares the previous fitness value, F_z^i of each parameter in $\Omega_z^{(I\times D)}$ with their current fitness value, F_{z+1}^i. If the current F_{z+1}^i is greater than F_z^i, then the NSG updates the current value. Furthermore, the NSG checks for the largest current F_{z+1}^i produced in the entire population of parameters, $\Omega_{z+1}^{(I\times D)}$ and updates this to determine the best (optimal) solution as Ω^{best}. Then, it uses Ω^{best} to compute γ^{best} by the FCME algorithm. A summary presentation of the NSG algorithm is provided in Algorithm 5.

4.4.5 ***Proposed heuristic for threshold estimation in the noise-only case*** We propose a heuristic algorithm called the noise-only detector (NOD) in order to guarantee the performance of the CSO-FCME model

Algorithm 4: Fitness Evaluation **(FE)**

Data:

1. The entire set (population) of PVs, $\Omega_z^{(I\times D)} = [T_{cme,z}, Q_z]$,
2. The offset input PSD samples, $S(n)$, $n = 1, 2, ..., N$,
3. The initial fitness function values, F_z^i, for $i = 1, 2, ..., I$, obtained using RISG
4. The lower and upper constraints, Lb, Ub.

Result:

1. A new set of fitness values, F_{z+1}^i, for $i = 1, 2, ..., I$,
2. A set of current threshold values, γ_{z+1}^i, $i = 1, 2, ..., I$, (i.e. the ETVs)
3. A new/current set of updated parameter values, $\Omega_{z+1}^{(I\times D)} = [T_{cme,z+1}, Q_{z+1}]$

Process:

Check that the set of parameter values (i.e. PVs) in $\Omega_z^{(I\times D)}$ do not exceed the stipulated constraints. In this case, constraints that do exceed the limits are forced to assume the respective values of the limits of each parameter. **for** *i = 1 to I* **do**

  for *d = 1 to D* **do**

    Obtain current parameter values as $\Omega_z^{i,d} = [T_{cme,z}^{i,d}, Q_z^{i,d}]$.

    Run the FCME algorithm (as in Fig. 3.2) in order to estimate new threshold values, γ_{z+1}^i, (i.e. the ETVs) based on each set of parameter values $T_{cme,z}^{i,d}$, and $Q_z^{i,d}$ (i.e. the PVs).

    Use γ_{z+1}^i to classify the input energy samples, $S(n)$, $n = 1, 2, ..., N$, into two classes namely, the noise set, $N_e^i(k)$, $k = 1, 2, ..., K$, where K is the total number of noise samples based on γ_{z+1}^i, and the signal set, $S^i(v)$, $v = 1, 2, ..., V$, where V is the total number of signal samples based on γ_{z+1}^i. The following holds: $N = K + V$.

    Compute the mean of the noise samples as $\mu_n^i = \frac{1}{K}\sum_{k=1}^{K} N_{e,k}^i$, the mean of the signal samples as $\mu_s^i = \frac{1}{V}\sum_{v=1}^{V} S_v^i$, the probability of the noise class as $p_n^i = \frac{K}{N}$, and the probability of the signal class as $p_s^i = \frac{V}{N}$, all computed based on the current threshold value, γ_{z+1}^i, estimated by the FCME algorithm.

    Compute the current fitness value, F_{z+1}^i for the current threshold value, γ_{z+1}^i, using $F_{z+1}^i = p_s^i \times p_n^i \times \left[\mu_s^i - \mu_n^i\right]^2$, (i.e. using eqn (4.1) and set the current parameters as $\Omega_{z+1}^i = [T_{cme,z+1}^i, Q_{z+1}^i]$.

  end

end

particularly in noise-only sample conditions. The NOD algorithm typically maintains the auto-tuning capability of our model in noise-only sample conditions. The NOD ensures that the CSO-FCME model does not misinterpret a noise-only spectrum as a signal-only spectrum,

Algorithm 5: New Solution Generator (NSG)

Data:

1. The new/current set of parameter values (i.e. PVs), $\Omega_{z+1}^{(I\times D)} = [T_{cme,z+1}, Q_{z+1}]$,
2. The previous fitness values of each PV, F_z^i, $i = 1, 2, ..., I$,
3. The current set of fitness values, F_{z+1}^i, $i = 1, 2, ..., I$ (computed by the **FE** function)
4. The lower and upper constraints, Lb, Ub.

Result:

1. The global best fitness value, $F^{\max}$, of the optimal PV
2. The global best parameter values (i.e. the PV), $\Omega^{best} = [T_{cme}^{best}, Q^{best}]$,
3. The global best threshold value (i.e. the FTV), γ^{best}.

Process:

for *i = 1 to I* **do**

  if $F_z^i \geq F_{z+1}^i$ **then**

    $F_{z+1}^i = F_z^i$ Update the previous fitness value to the current fitness value only if the condition in step 2 is true

    $\Omega_{z+1}^{(i\times D)} = \Omega_z^{(i\times D)}$ Update the previous parameter values to the current pair of values only if step 2 is true

  else

    Proceed to Step 9

  end

end

Compute the current fitness value, F_{z+1}^i for the current threshold value, γ_{z+1}^i, using $F_{z+1}^i = p_s^i \times p_n^i \times \left[\mu_s^i - \mu_n^i\right]^2$, (i.e. using eqn (4.1)) and set the current parameters as $\Omega_{z+1}^i = [T_{cme,z+1}^i, Q_{z+1}^i]$.

Obtain global best pair of parameter values, $\Omega^{best} = [T_{cme}^{best}, Q^{best}]$, corresponding to $F^{\max}$.

Obtain best threshold, γ^{best}, corresponding to $F^{\max}$.

The final threshold value is rescaled as $\gamma^{best} = \gamma^{best} + Y_{\min}$ (recall the offset function in Eq. 4.4)

as this may lead to higher P_{FA} rates. Furthermore, the NOD ensures that it is no longer necessary to manually predefine a target P_{FA} for the FCME algorithm, instead, our model can automatically and accurately determine an optimal P_{FA} rate only by processing the input dataset, $Y(n)$ measured per time.

The NOD works as follows: It measures the degree of closeness between the estimated threshold value, γ^{best}, and the mean of the entire dataset, μ_T. If the difference is smaller than 10% of the entire sample range, and the γ^{best} is less than or equal to μ_T, then it declares the sample set as being noise-only samples. The algorithm then estimates a final threshold value by examining the edges of the band where it assumes to find noise samples. The NOD assumes that the presence of signal samples in the dataset typically causes a bias toward higher threshold values. Consequently, the smaller the bias, the

Algorithm 6: Noise-only detector (NOD)

Data:
1. The optimal threshold value, γ^{best}, estimated by the **NSG** algorithm
2. The original input energy samples, $Y(n)$, $n = 1, 2, ..., N$

Result:
1. The final threshold value, γ^{best}

Process:

Compute the mean of the energy samples as $\mu_T = \frac{1}{N}\sum_{n=1}^{N} Y_n$.
if $\lceil \gamma^{best} \rceil \leq \lceil \mu_T \rceil$ **then**
  if $\left|\mu_T - \gamma^{best}\right| \leq 0.1 * [\max(Y(n)) - \min(Y(n))]$, *for* $n = 1, 2, ..., N$ **then**
    $\theta = 0.1 \times N$
    $\gamma^{best} =$ $\min\left[\max_{n=1,2...,\theta}(Y(n)),\ \max_{n=N-\theta,N-\theta+1,...,N}(Y(n))\right]$
  else
    $\gamma^{best} = \gamma^{best}$ % The final threshold value remains unchanged if the above conditions are not met
  end
else
  $\gamma^{best} = \gamma^{best}$ % The final threshold value remains unchanged if the above conditions are not met
end

more likely the sample set contains noise-only samples. A summary of the NOD is provided in Algorithm 6.

5 Empirical method of analysis

We evaluated our CSO-FCME model and compared it with existing methods using the probability of detection, P_D, and the false alarm probability, P_{FA} statistically described as

$$P_D = \Pr(Y(n) > \gamma \mid H_1), \qquad n = 1, 2, ..., N \quad (5.1)$$
$$P_{FA} = \Pr(Y(n) > \gamma \mid H_0), \qquad n = 1, 2, ..., N \quad (5.2)$$

where $Y(n)$ denotes the signal spectra values for different frequency index, $n = 1, 2, ..., N$, and γ is the threshold value typically estimated by the model, H_0 is the null hypothesis, which states that there exist no signal samples in the dataset (noise-only spectra), while H_1 is the alternate hypothesis, which states that there exist both signal and noise samples in the dataset. P_D and P_{FA} were computed following Fawcett's empirical approach [49].

Fawcett's approach relates to our work based on the illustration in Fig. 5.1 [46]. Based on Fig. 5.1, we labeled each test dataset in a binary approach to obtain the ground truth. The ground truth of each test dataset was labeled using the true threshold value, γ_G of the dataset (see Fig. 5.1). We used the thermal noise floor of the detector as the true threshold value, γ_G. Thus, any sample value greater than γ_G was labeled as 1 (true signal), while any sample value less than or equal to γ_G was labeled as 0 (true noise). This binary labeling approach was used to generate the ground truth of each use-case dataset considered in our experiments. The different datasets used are presented and discussed in Section 6.

Likewise, our model and other methods were subjected to each dataset in order to estimate a threshold value. To analyze their respective performances, we computed the P_D and P_{FA} as follows: we declared a missed detection if the sample in the ground truth was labeled as 1, and the same sample was labeled by the threshold estimation algorithm as 0 (see Fig. 5.1a). We declared a false alarm if the sample in the ground truth was labeled as 0 and the same sample was labeled by the threshold algorithm as 1 (false positive (see Fig. 5.1b)). Furthermore, we declared a correct detection when a sample in the ground truth was labeled as 1 and the same sample was labeled by the threshold algorithm as 1 (i.e. a true positive).

Following this analysis, we then computed the P_D per dataset as [49]:

$$P_D = \frac{\phi}{\rho}, \quad (5.3)$$

where ϕ denotes the total number of true positives (truly detected signal samples) if $Y(n) > \gamma \mid H_1$, and ρ is the total number of actual true signal samples (obtained by summing the total number of ones in the ground truth). We calculated the P_{FA} as well using

$$P_{FA} = \frac{\varphi}{\eta}, \quad (5.4)$$

where φ denotes the total number of false positives (falsely detected signal samples) if $Y(n) > \gamma \mid H_0$, and η is the total number of noise samples (obtained by summing the total number of zeroes in the ground truth). Then, based on the empirically graphed receiver operating characteristic (ROC) curve (that is, P_D vs P_{FA}) (see [49]), we obtained and tabulated the point performances of each algorithm as in Section 6.

6 Results and Discussion

First, we present our findings concerning why we used the CSO algorithm in our model. To achieve this, we developed and compared our model considering different metaheuristic algorithms namely the CSO, artificial bee colony (ABC) [50], particle swarm optimization (PSO) [51], genetic algorithm (GA) and the differential evolution (DE) algorithm [52]. We analyzed the speed (number of fitness evaluations consumed before convergence) and the accuracy (highest fitness value) of these algorithms followed by the average processing time of

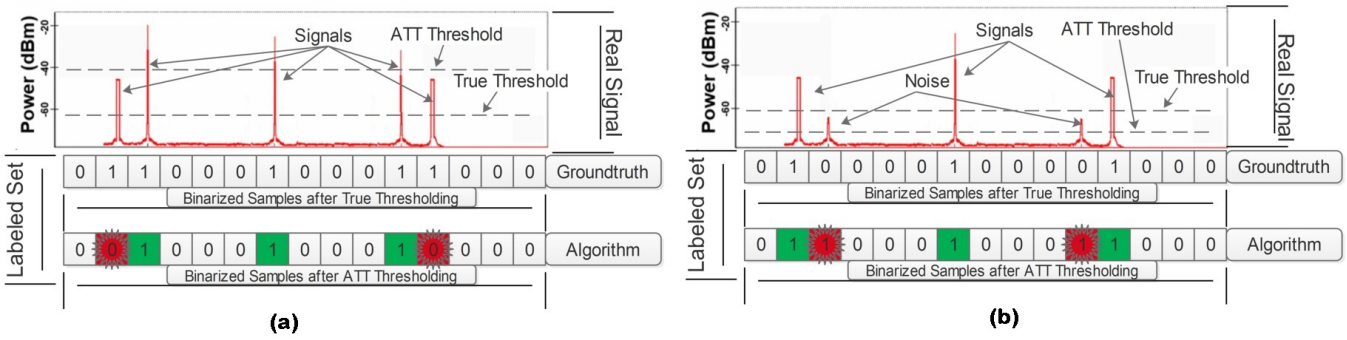


Fig. 5.1: Method of Labeling and Analysis showing different errors: (a) Missed Detection, and (b) False alarms

our CSO-FCME model compared to the baseline FCME algorithm.

In subsequent subsections, we analyzed our CSO-FCME model and the baseline FCME algorithm under two phases namely, the parameter-tuning and testing phases. In the parameter-tuning phase, we carefully adjusted the parameters of the baseline FCME algorithm to determine its best operating values. In the testing phase, the parameters were kept fixed and compared with our model. Usually, this would be the case in real-life deployments where the baseline FCME algorithm typically cannot auto-change its parameter values. Furthermore, in the testing phase, each algorithm was tested with datasets different from the datasets used in the parameter-tuning phase. We analyzed and discussed the results obtained considering the IEEE 802.22 standard, which requires a minimum $P_D > 90\%$ and a maximum $P_{FA} < 10\%$ [53]. Then, we compared our model with other state-of-the-art methods such as the MED-FCME and a fully autonomous method such as the modified Otsu's algorithm (MOA). We also tested each algorithm considering different channel conditions including Raleigh fading channels. All datasets used in our research are made freely accessible in [54]. All experiments and results reported in our paper were encoded and simulated using MATLAB 2017a.

6.1 ***Choice of Metaheuristic Algorithm***

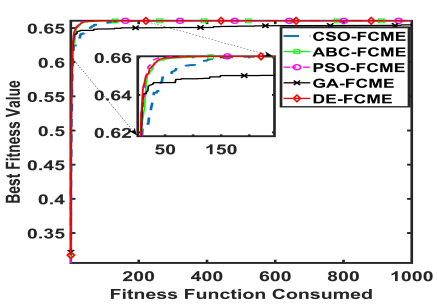
Table 6.1 provides the parameter values used to run the different metaheuristic algorithms compared and analyzed in the present paper. These values were obtained following several trials that guaranteed the best performance of our model. Fig. 6.1 presents the convergence trend of each algorithm per fitness evaluation consumed. We used frequency modulated (FM) and orthogonal frequency division multiplexing (OFDM) signals in our experiments since these signals are often encountered by CR systems within the TV spectra. In each case, we considered the signal-to-noise ratio (SNR) of each signal under both poor ($SNR = 1dB$) and good ($SNR = 10dB$) signal conditions. The FFT sample length of both the FM and OFDM signals were kept fixed at 250 samples each, respectively. The results obtained were based on a stopping criterion of 1000 fitness functions consumed, population size = 10, and the values reported were averaged over 100 Monte Carlo trials in order to minimize statistical uncertainties.

The CPU processing time of each algorithm was measured based on a PC running an Intel(R) Core i5-7500 CPU processor @ 3.40 GHz with an installed memory (RAM) of $16GB$. Our findings reported in Table 6.2 - 6.5 present the average CPU processing time over 100 different Monte Carlo trials and the best fitness value obtained after 100 fitness functions consumed. To guarantee the validity of our results, the following experimental conditions were noted: (1) All simulations were conducted using MATLAB 2017a. (2) All code lines within each algorithm were suppressed to minimize any extra processing time that may be incurred. (3) The population size and number of fitness function consumed were kept fixed across all algorithms. (4) We began measuring the processing time strictly before the *Initialization* of each algorithm and we ended measurement strictly after the algorithm completes its main loop. (5) Only MATLAB software was kept running in the PC during the experimental process in order to minimize any extra processing cost that may be incurred. (6) We averaged our results per algorithm over 100 independent Monte Carlo trials. Essentially, our aim was to examine the timing and accuracy (best fitness value) performance of each algorithm assuming that the different algorithms were allowed to consume the same amount of fitness function (=1000).

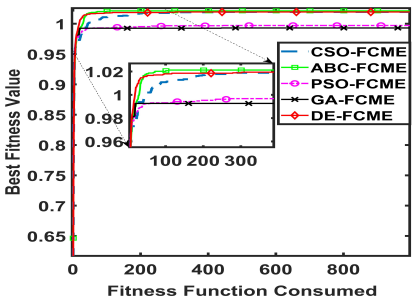
Closer examination of Figs. 6.1 (a) - (d) indicates that the ABC and DE algorithms often set out as the quickest algorithms towards converging to the best fitness value. Nevertheless, further detailed analyses as presented in Tables 6.2 - 6.5 suggest that the CSO algorithm often achieved the best fitness value in most cases within the shortest physical CPU processing time. Apart from the best accuracy performance achieved by the ABC algorithm in Table 6.3, we note that the CSO algorithm often achieved the best performance in terms of CPU processing time and overall best fitness value. The real-

Table 6.1: Parameter settings of the different Metaheuristic algorithms used in our experiments (population size = 10, Number of fitness function consumed = 1000, Monte Carlo trials = 100)

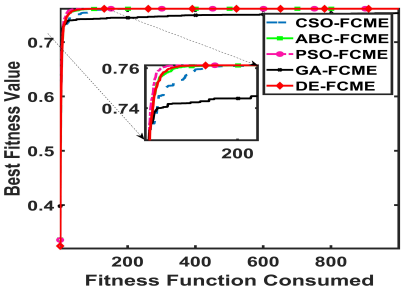
Algorithms	**Parameter Settings**
CSO	Discovery rate of Alien eggs = 0.2
ABC	Number of Onlooker Bees = 10 Abandoment (Trial) limit = 12 Acceleration Coefficient (upper bound) = 1
PSO	Self adjustment weight = 1.5 Social adjustment weight = 1.5 Inertia weight (adaptive) = (0.1,1.1) Acceleration coefficients = Uniformly distributed (0,1) random vector
GA	Crossover rate = 0.8 Mutation rate = 0.2 Selection rate = 0.8 Selection operator = Roullette wheel Crossover type = Single point crossover Chromosome lenght = 16 Elitism rate = 0.5
DE	Crossover probability = 0.5 Differential weight = 0.8



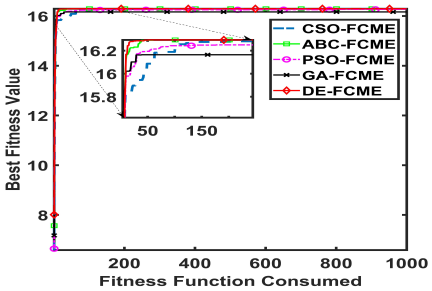
(a) FM signal condition at SNR = 1 dB



(b) FM signal condition at SNR = 10 dB



(c) OFDM signal condition at SNR = 1 dB



(d) OFDM signal condition at SNR = 10 dB

Fig. 6.1: Fitness performance of different Metaheuristic algorithms under different signal conditions

Table 6.2: Detailed performance Analysis of the different Metaheuristic algorithms for FM Signal Condition at SNR = 1dB (population size = 10, Number of fitness function consumed = 1000, Monte Carlo trials = 100)

Algorithms	**CPU Time (Secs)**	**Best Fitness Value**
CSO	**0.0389 ± 0.0154**	**0.6606**
ABC	0.0613 ± 0.0227	**0.6606**
PSO	0.0609 ± 0.0184	**0.6606**
GA	0.0927 ± 0.0282	0.6547
DE	0.0586 ± 0.0098	**0.6606**

Table 6.3: Detailed performance Analysis of the different Metaheuristic algorithms for FM Signal Condition at SNR = 10dB (population size = 10, Number of fitness function consumed = 1000, Monte Carlo trials = 100)

Algorithms	CPU Time (Secs)	Best Fitness Value
CSO	**0.0388 ± 0.0105**	1.0194
ABC	0.0616 ± 0.0192	**1.0213**
PSO	0.0598 ± 0.0122	0.9971
GA	0.0971 ± 0.0190	0.9930
DE	0.0571 ± 0.0103	1.0194

Table 6.4: Detailed performance Analysis of the different Metaheuristic algorithms for OFDM Signal Condition at SNR = 1dB (population size = 10, Number of fitness function consumed = 1000, Monte Carlo trials = 100)

Algorithms	CPU Time (Secs)	Best Fitness Value
CSO	**0.0387 ± 0.0109**	**0.7615**
ABC	0.0573 ± 0.0173	**0.7615**
PSO	0.0567 ± 0.0129	**0.7615**
GA	0.0880 ± 0.0191	0.7538
DE	0.0583 ± 0.0119	**0.7615**

Table 6.5: Detailed performance Analysis of the different Metaheuristic algorithms for OFDM Signal Condition at SNR = 10dB (population size = 10, Number of fitness function consumed = 1000, Monte Carlo trials = 100)

Algorithms	CPU Time (Secs)	Best Fitness Value
CSO	**0.0397 ± 0.0124**	**16.2931**
ABC	0.0571 ± 0.0158	**16.2931**
PSO	0.0576 ± 0.0128	16.2677
GA	0.0889 ± 0.0209	16.1643
DE	0.0611 ± 0.0091	**16.2931**

coded GA algorithm typically fell short in most use-cases, followed by the PSO algorithm (see Tables 6.2 - 6.5). Summarily, we note that the ABC and DE algorithm suffice as very impressive performers and they are potential candidates for use in our model. However, for the purpose of detecting signals autonomously and accurately in Cognitive Radio applications, we examine the CSO algorithm further in the rest of our experiments.

6.2 ***Parameter-tuning Phase:*** *The H_0 condition*

6.2.1 ***In Uniform Noise-only Condition*** We used randomly generated additive white Gaussian noise (AWGN) sample sets ($N = 250$) in the noise-only sample condition to determine whether one of the main control parameters of the CSO algorithm (i.e. the probability of discovering alien eggs, P_a) will greatly affect the P_{FA} performance of our CSO-FCME model or not. To conduct this and subsequent experiments, we iterated the CSO algorithm using $I = Z = 5$, where Z is the total number of iterations by the CSO algorithm and I is the number of nests (i.e. candidate parameter solutions, which we called PVs in Fig. 4.1).

We also note that the following constraints were used during the optimization process: $0.1 \leq T_{cme} \leq 5$ and $1 \leq Q \leq 25$ (recall Section 4.3). Using these values in our experiments made the CSO-FCME model to converge quickly to optimal solutions. We report the result of our simulations in Fig. 6.2, which shows that a fairly constant P_{FA} was maintained despite using different P_a values for the CSO algorithm (i.e. $0 < P_a \leq 1$). Consequently, we used $P_a = 0.2$ in our subsequent experiments even though any P_a may still be suitable for the CSO-FCME model. We used $P_a = 0.2$ because it produced the least average P_{FA}. An interesting point to note is that our CSO-FCME model auto-tuned itself in order to achieve the low P_{FA} rate ($P_{FA} < 0.04$) reported in Fig. 6.2. This automatic and effective capability of our model contributes mainly to adaptive threshold estimation in CR since it makes it possible to achieve fully blind spectrum sensing.

Next, we used Monte Carlo simulation (considering 100 different trials) to carefully tune the baseline FCME algorithm using AWGN noise-only sample sets. We present the averaged result in Fig. 6.3, which shows the average threshold value estimated by the carefully tuned baseline FCME algorithm and our CSO-FCME model. Both algorithms achieved $P_{FA} < 0.01$, which can be easily corroborated by the thresholds being above the noise level in Fig. 6.3. An important note is that while we carefully tuned the baseline FCME algorithm over 100 different Monte Carlo trials to improve its accuracy, instead, our CSO-FCME model automatically converged to a similar accurate threshold value after $Z = 5$ iterations. This

Table 6.6: Probability of false alarm performance under noise uncertainty conditions

Noise Uncertainty δ **(dB)**	**Baseline FCME Algorithm** P_{FA}	**CSO-FCME Model** P_{FA}	**Threshold Factor** (T_{cme})	**Initial Clean Set,** Q (%)	**Threshold (in dBm)**
0	0.0300	1.0000e-3	1.9159	19.5000	-97.4859
1	0.0200	1.0000e-3	2.4697	9.7700	-96.6800
2	0.0240	1.0000e-3	2.7487	5.3500	-96.0310
3	1.0000e-3	1.0000e-3	2.9334	4.0400	-96.0470
4	1.0000e-3	1.0000e-3	2.0297	12.8000	-94.7330
5	0.3200	1.0000e-3	3.5090	9.2500	-93.8376

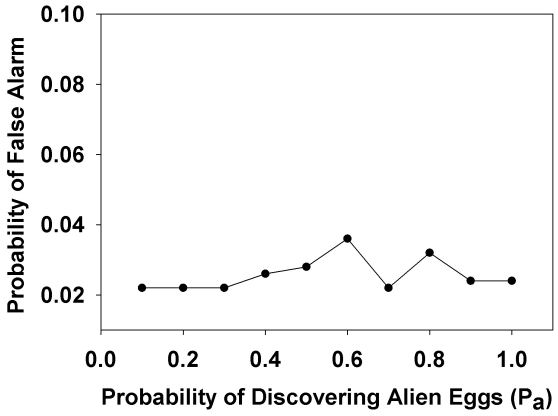


Fig. 6.2: False alarm Rate of the CSO-FCME model based on different discovery rate of the alien eggs within the CSO algorithm

quick, accurate and automatic convergence of our model is an interesting contribution of our research. Please note that the result of Fig. 6.3 was obtained using the following carefully tuned parameter values of the baseline FCME algorithm: $T_{cme} = 4.1$ and $Q = 10\%$, in order to maintain a target $\widehat{P}_{FA} = 0.05$. Since these values of the baseline FCME algorithm cannot be easily changed in real-time usage, we then kept them fixed in subsequent test conditions in order to mimic real-life conditions.

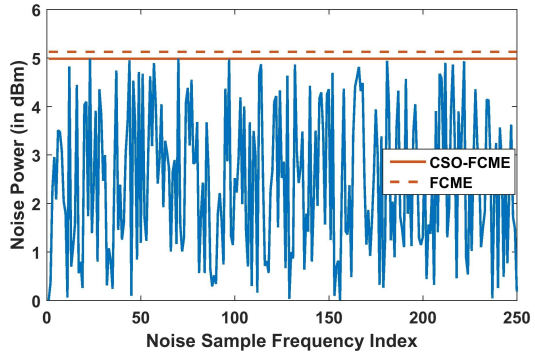


Fig. 6.3: Average estimated threshold by the CSO-FCME and baseline FCME algorithm under the uniform AWGN noise-only case

6.2.2 ***Under noise uncertainty conditions*** We studied the important effect of noise uncertainty on the performance of the baseline-FCME algorithm and the CSO-FCME model. We increased the noise uncertainty level, δ (dB) in the AWGN noise set by gradually adding random numbers drawn from a Uniform distribution of same length in the range $(0, \delta)$ in steps of $1dB$. In this case, we kept the parameters of the baseline FCME algorithm fixed at $T_{cme} = 4.1$ and $Q = 10\%$ in order to maintain a target $\widehat{P}_{FA} = 0.05$. The noise sample length was kept at $N = 250$. These values were defined and kept constant for all noise uncertainty levels. We report our results in Table 6.6.

From Table 6.6, while we varied the noise uncertainty level, we observed that the CSO-FCME model automatically varied its T_{cme} and Q values to estimate accurate threshold values to maintain a low P_{FA} rate (see Table 6.6). The CSO-FCME model performed better than the baseline FCME algorithm because of the NOD algorithm, which ensured that the noise-only condition was always identified and threshold values were estimated above the noise level. We have provided the datasets used to perform these noise uncertainty experiments in [54] and the estimated threshold and parameter values are provided in Table 6.6 for easier verification. Summarily, Table 6.6 shows that the CSO-FCME model outperformed the baseline FCME algorithm particularly under large noise uncertainty levels making our model suitable for use in CR.

6.3 ***Test phase: The*** H_1 ***condition***

6.3.1 ***To determine minimum SNR level*** The CSO-FCME and the baseline FCME algorithm were examined under different SNR levels. These experiments aimed to determine the minimum SNR level below which the performance of the CSO-FCME and the baseline FCME algorithm may no longer be guaranteed. We used the same parameter values from the parameter-tuning phase to configure the baseline FCME algorithm, i.e. the results

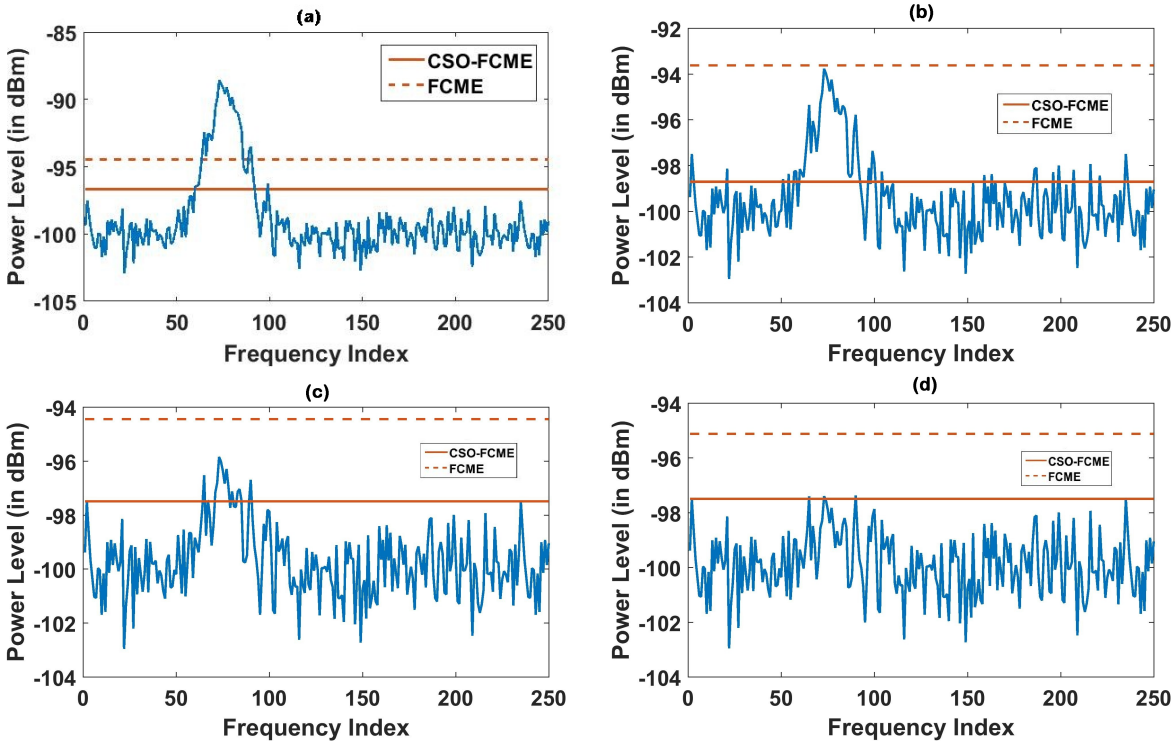


Fig. 6.4: Threshold Computation for CSO-FCME and Unoptimized FCME in (A) SNR = 10dB, and (B) SNR = 5dB (C) SNR = 3dB (D)SNR = 1dB

of the baseline FCME algorithm in the test phase were based on $T_{cme} = 4.1$, and $Q = 10\%$.

On the other hand, the parameters of the CSO-FCME were computed automatically and adjusted by the model per input dataset. The different signals and the corresponding estimated threshold values are shown in Figs. 6.4a - d. The $SNR = 10dB$ use-case is shown in Fig. 6.4a for which the CSO-FCME algorithm produced a detection rate of $P_D = 100\%$ and a false alarm rate of $P_{FA} = 0.01\%$, while the baseline FCME algorithm achieved $P_D = 72.73\%, P_{FA} = 0\%$. It is seen in Figs. 6.4b - d that the CSO-FCME model provides a better detection performance than the baseline FCME. We see that even though a zero false alarm rate was maintained by the baseline FCME algorithm for all SNR levels, however, this was achieved at the expense of a lower probability of detection. The poorer detection performance of the baseline FCME algorithm was traced to its inability to autonomously adapt its parameter values based on the different use-cases.

Thus, the results of Fig. 6.4a - d typically support the need for self-configurability as demonstrated by our proposed CSO-FCME model. The CSO-FCME model performed well in conditions as low as $SNR = 3dB$, below which its detection performance may no longer be guaranteed. The baseline FCME algorithm only performed well above $SNR = 5dB$, below which its performance also degraded.

6.3.2 ***Effects of different sample lengths*** The effect of short and long sample lengths on the performance of both algorithms was analyzed. This experiment reveals the possible effect of fast and slow sensing times on the performance of both algorithms. In this case, the high SNR level use-case was considered in order to ensure that both algorithms were capable of detecting the signal. Fig. 6.5a shows a band with sample length $N = 250$. The same spectrum was maintained as in Fig. 6.5b albeit shorter sample lengths of $N = 125$, and $N = 75$ as in Fig.6.5c. By considering half of the sample length, the baseline FCME algorithm is observed to estimate higher threshold values as compared to the CSO-FCME model, which estimated more averagely constant threshold values over the different sample lengths. It is believed that the reduction in sample length may have caused the baseline FCME algorithm to consider more signal samples during the estimation process, thus estimating higher threshold values. Different from the baseline FCME algorithm, our self-configurable model successfully adjusted its parameter values in order to estimate more accurate threshold values for the different sample lengths. The different threshold values and their corresponding parameter values are presented for each sample length in Table 6.7. The different parameter values estimated by our model are shown in Table 6.7 to support the self-configurability capability of our model.

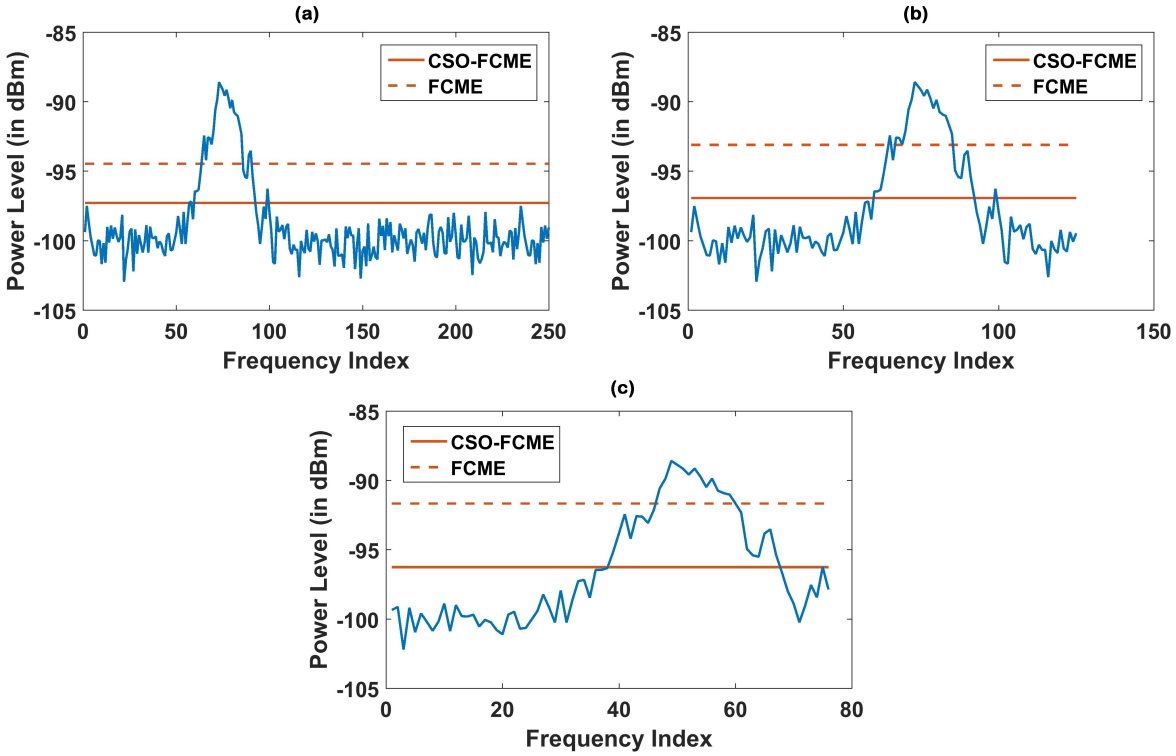


Fig. 6.5: Threshold Computation for CSO-FCME and Unoptimized FCME in Sample Lengths of (A) N = 250 samples, (B) N = 125 samples, and (C) N = 75 samples

Table 6.7: Detection Performance in different Sample Length Conditions

Algorithms	Sample Lengths	Threshold Factor (T_{cme})	Initial Clean Set (%)	Threshold (in dBm)	P_D(%)
CSO-FCME	250	2.332	17.55	-97.2697	94.74
	125	2.501	4.26	-96.9205	91.89
	75	2.338	22.46	-96.2426	80.56
Base-line FCME	250	4.1	10	–94.4498	63.16
	125	4.1	10	-93.0902	54.05
	75	4.1	10	–91.6484	38.89

6.3.3 ***Effects of low and high occupancy rates*** The effect of different signal occupancy rates on the performance of both algorithms was examined. The results obtained in these use-cases are shown in Fig 6.6. Our model is shown to adjust its parameter values in accordance with the different occupancy rates considered in our experiment. Nevertheless, it is shown to perform poorly for the 100% occupancy rate case. This poor detection performance in Fig. 6.6d is attributed to the large number of signal samples in the initial clean sample set (ICSS) particularly for the 100% occupancy rate condition.

Nevertheless, the CSO-FCME algorithm is shown in the results of Table 6.8 to provide better detection performance as compared to the baseline FCME algorithm. It is worth noting that a $P_{FA} \approx 0\%$ was recorded for both algorithms as easily seen in the plots of Figs. 6.6a - d in which the threshold lines are above the noise level.

6.4 ***Comparison with other state-of-art techniques***

In this section, we compare our model with the median filtered FCME (MED-FCME) and the modified Otsu's algorithm (MOA). We considered the MOA because it is a notable and fully autonomous algorithm [13]. The MED-FCME is a notable enhancement to the FCME algorithm, which makes it stable in conditions with high noise variance. However, the MED-FCME also required us to fine-tune its median filter length, M, which we did under Rayleigh fading conditions. We used the same parameter settings for the baseline FCME and MED-FCME algorithm as follows: $T_{cme} = 4.1$ and $Q = 10\%$. We tested each algorithm under Raleigh fading conditions at $SNR = 5$ and $10dB$, respectively and discuss the results obtained in the following subsections:

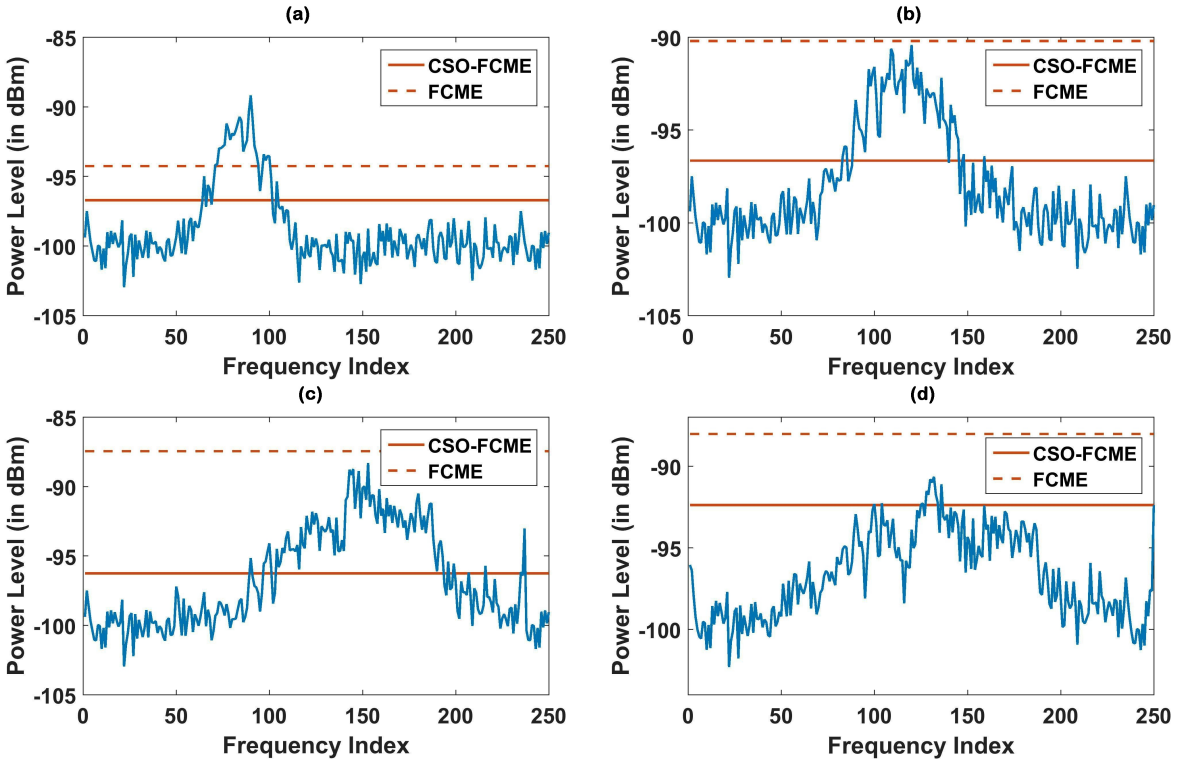


Fig. 6.6: CSO-FCME and Unoptimized FCME under different Occupancy Rates (in Percentage) of (A) 25, (B) 50, (C) 75, (D) 100

Table 6.8: Detection Performance under different Occupancy Rates

Algorithm	**Occupancy (%)**	**Threshold Factor** (T_{cme})	**Initial Clean Set , Q(%)**	**Threshold (in dBm)**	**P$_D$(%)**
CSO-FCME	25	2.253	5.19	-96.6977	78.26
	50	2.093	7.81	-96.6390	81.33
	75	1.9981	10.73	-96.2393	87.93
	100	1.7833	13.37	-92.3730	8.88
Base-line FCME	25	4.1	10	-94.2515	60.87
	50	4.1	10	-90.1957	0
	75	4.1	10	-87.4451	0
	100	4.1	10	-88.02	0

Table 6.9: Performance under Rayleigh faded condition at SNR = 5dB

Algorithm	**Threshold Factor** (T_{cme})	**Initial Clean Set** (%)	**Threshold (in dBm)**	**P$_D$(%)**	**P$_{FA}$(%)**
CSO-FCME	2.1425	12.18	-74.7040	90.48	0.001
Base-line FCME	4.1	10	-68.2622	4.76	0.001
MED-FCME ($M = 5$)	4.1	10	-78.3218	100	22.39
MED-FCME ($M = 10$)	4.1	10	-77.8279	100	19.40
MED-FCME ($M = 20$)	4.1	10	-80.7516	100	50.25
MED-FCME ($M = 50$)	4.1	10	-81.5117	100	53.73
MOA	-	-	-81.7667	100	60.20

6.4.1 ***Under Rayleigh Fading Condition*** We simulated Rayleigh fading channels based on a 9600 sample rate and a maximum Doppler shift of 100 Hz to examine the performance of each algorithm. We considered two different channel conditions of $SNR = 5$ and $15dB$ to model both low and high SNR conditions, respectively.

Table 6.10: Performance under Rayleigh faded condition at SNR = 15dB

Algorithm	**Threshold Factor** (T_{cme})	**Initial Clean Set** (%)	**Threshold (in dBm)**	$\mathbf{P}_D(\%)$	$\mathbf{P}_{FA}(\%)$
CSO-FCME	3.0626	17.60	-71.7164	90.1	0.001
Base-line FCME	4.1	10	-67.2810	80.0	0.001
MED-FCME ($M = 5$)	4.1	10	-77.3700	100	14.84
MED-FCME ($M = 10$)	4.1	10	-77.6902	100	19.23
MED-FCME ($M = 20$)	4.1	10	-80.7516	100	47.80
MED-FCME ($M = 50$)	4.1	10	-81.5117	100	50.55
MOA	-	-	-71.4000	90.0	0.001

The results obtained for both SNR levels are presented in Tables 6.9 and 6.10, respectively. In each condition, we compared our model with the MOA and different parameter settings of the MED-FCME algorithm. Following the results obtained in Table 6.9, we observed the following: our CSO-FCME model successfully auto-tuned its parameter values in order to estimate a better threshold value that ensured $P_D = 90.48\%$ and $P_{FA} = 0.001\%$. The baseline-FCME and MED-FCME algorithms at different median lengths achieved very high P_D rates at the expense of very high P_{FA} rates (see Table 6.9). The MOA also achieved similar high P_D and P_{FA} rates. In this case, only the CSO-FCME model satisfied the IEEE 802.22 standard (i.e. $P_D > 90\%$, $P_{FA} < 10\%$), while the other algorithms did not. This performance supports the viability and effectiveness of our proposed model.

Considering the high SNR case in Table 6.10, we observed that our CSO-FCME model and the MOA achieved $P_D = 90.1\%$ and $P_{FA} = 0.001\%$, which satisfied the IEEE 802.22 standard. However, the baseline-FCME algorithm and the MED-FCME at different median lengths did not satisfy the IEEE 802.22 standard. Though the MED-FCME and baseline FCME algorithms achieved very high P_D rates, they however suffered from similar high P_{FA} rates. We also provided the T_{cme} and Q parameter values of the CSO-FCME model in Table 6.10 to demonstrate the auto-tuning capability of our model. It is interesting to note that our model successfully and automatically fine-tuned the FCME algorithm's parameters quite accurately. The dataset used to conduct these experiments are again provided in [54].

6.4.2 ***Under an unseen spectra condition*** We used the same settings in Section 6.4.1 to examine each algorithm based on an unknown dataset, which is an important case that occurs frequently in real-life conditions. The distribution of this spectrum was unknown to the different algorithms, thus making it worthwhile in our investigation since most distributions will typically be unknown in real-life situations. We present the spectra in Fig. 6.7, where the signal lies between the 390^{th} - 590^{th} frequency sample while all other samples were strictly noise-only samples. In this case, each algorithm maintained $P_{FA} < 5\%$. Following our analysis presented in Table 6.11, we again show that our model provided a better P_D rate than the other methods. In particular, the results presented thus far suggest that our model effectively auto-tunes the parameters of the FCME algorithm, which forms a significant contribution of our research.

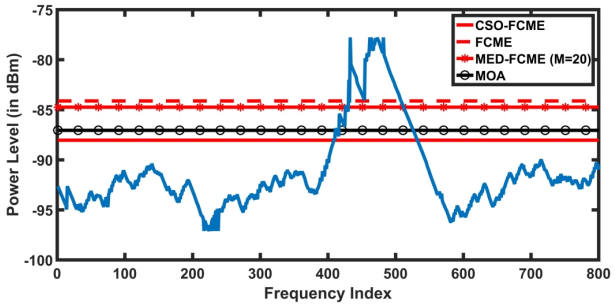


Fig. 6.7: Spectra assumed to be obtained from an unknown distribution along with the estimated threshold values by the different algorithms

6.5 ***Timing Performance of the CSO-FCME Model***

This section reports the timing performance of our CSO-FCME model compared with the baseline FCME algorithm. This performance evaluation was considered necessary since our model obviously introduces an extra processing cost to the baseline FCME algorithm. For this purpose, our experiments in this regard were conducted using a PC running an Intel(R) Core i5-7500 CPU processor @ 3.40 GHz with an installed memory (RAM) of $16GB$. We used the worst case scenario of $SNR = 1dB$ for the OFDM signal to analyze the timing performance of both algorithms. Our findings are reported in Table 6.12 showing each processing time obtained over an average of 100 Monte Carlo simulations considering different population sizes (I) and number of iterations (Z). We note that our model achieves its minimum timing

Table 6.11: Performance under an unseen spectra condition (obtainable in the real-world deployments)

Algorithm	**Threshold Factor** (T_{cme})	**Initial Clean Set** (%)	**Threshold (in dBm)**	$\mathbf{P}_D(\%)$	$\mathbf{P}_{FA}(\%)$
CSO-FCME	2.2640	12.17	-88.93	92.36	0.01
FCME	4.1	10	-84.12	55.05	0.01
MED-FCME ($M = 20$)	4.1	10	-84.75	59.72	0.01
MOA	-	-	-87.06	82.65	0.01

Table 6.12: Timing performance of the CSO-FCME algorithm

		Processing Time (Secs)		
Algorithms	**Population Size** (I)	Iteration (Z) = 5	Iteration (Z) = 50	Iteration (Z) = 100
CSO-FCME	5	0.0047	0.0292	0.0558
	10	0.0069	0.0519	0.0973
	20	0.0114	0.0932	0.1760
Baseline FCME	-	1.0889e-05		

performance considering a combination of $I = 5$ and $Z = 5$. Table 6.12 shows that the processing time can be increased using larger I and Z values with possibilities for better performance albeit longer processing time. Although our model obviously consumes longer processing time than the baseline FCME alogrithm, nevertheless, Table 6.12 confirms that the CSO-FCME model computes within the timing performance required by the IEEE 802.22 standard, which stipulates a maximum sensing period of 2 Secs for CR systems [53].

7 Conclusion

In this paper, we have proposed a new model that accurately auto-tunes the parameters of the forward consecutive mean excision (FCME) algorithm based only on the input dataset under consideration per time. Previously, users of the FCME algorithm are required to manually compute and fine-tune the algorithm's parameters in order to maintain a pre-defined P_{FA} rate. However, this approach can be prone to errors, which may lead to wrongly estimated threshold values causing either increased interference to the primary user (PU) or decreased spectra utilization by the cognitive radio (CR) user. Furthermore, this manual parameter tuning approach makes it difficult to automatically adjust the parameters of the FCME algorithm under dynamic spectra conditions. Consequently, we have introduced in the present paper a new model that addresses this limitation. Our goal was achieved by adapting the cuckoo search optimization (CSO) algorithm and the between-class variance function used in Otsu's algorithm to automatically search for the optimal parameter values of the FCME algorithm per input dataset. We chose the CSO algorithm based on its relatively better performance over some other notable metaheuristic algorithms such as the particle swarm optimization (PSO), artificial bee colony (ABC), genetic algorithm (GA), and differential evolution (DE) algorithms. We also developed a new heuristic algorithm to auto-tune the P_{FA} performance of our model under typical noise-only conditions. This heuristic allows our model to quickly and accurately adjust the parameters of the FCME algorithm in order to maintain its P_{FA} performance. Experiments were conducted to demonstrate that our CSO-FCME model performs better than the baseline FCME algorithm under different possible CR working conditions. Our model maintained good performance under both fast and slow sensing conditions. It has been shown to be effective at SNR levels down to $SNR = 3dB$ and under different occupancy rates ($\approx 75\%$ signal occupancy rates). However, since our model is obviously more complex than the baseline FCME algorithm, it thus computes threshold values in a longer period of time (i.e. slower speed) than the baseline algorithm. Furthermore, it may be limited in terms of its detection rate under high occupancy conditions ($> 75\%$), which is an issue to be considered in future works. However, based on the results presently obtained, it is sufficient to conclude that our model achieves fully blind spectrum sensing in CR with an additional potential to convert other highly parameterized adaptive threshold algorithms into effective and fully autonomous algorithms.

8 Compliance with Ethical Standards

8.1 ***Conflict of Interest***

H. Abdullahi declares that he has no conflict of interest. A. J. Onumanyi declares that he has no conflict of interest. S. Zubair declares that he has no conflict of interest. A. M. Abu-Mahfouz declares that he has no conflict of interest. G. P. Hancke declares that he has no conflict of interest.

8.2 ***Ethical approval***

This article does not contain any studies with human participants or animals performed by any of the authors

References

1. L. Arienzo and D. Tarchi, "Statistical modeling of spectrum sensing energy in multi-hop cognitive radio networks," *IEEE Signal Processing Letters*, vol. 22, pp. 356–360, Mar. 2015.
2. C. Liu, M. Li, and M.-L. Jin, "Blind energy-based detection for spatial spectrum sensing," *IEEE Wireless Communications Letters*, vol. 4, pp. 98–101, feb 2015.
3. G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges," *IEEE Access*, vol. 3536, no. c, 2017.
4. S. S. Oyewobi and G. P. Hancke, "A survey of cognitive radio handoff schemes, challenges and issues for industrial wireless sensor networks (CR-IWSN)," *Journal of Network and Computer Applications*, vol. 97, pp. 140–156, Nov. 2017.
5. I. F. Akyildiz, W. Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, pp. 2127–2159, 2006.
6. S. D. Barnes and B. T. Maharaj, "Prediction based channel allocation performance for cognitive radio," *AEU - International Journal of Electronics and Communications*, vol. 68, pp. 336–345, Apr. 2014.
7. K. Ntshabele, B. Isong, and A. M. Abu-Mahfouz, "Analysis of energy inefficiency challenges in cognitive radio sensor network," in *in the 44th Annual Conference of the IEEE Industrial Electronic Society*, (Washington D. C., USA), Oct. 2018.
8. F. Wasonga, T. O. Olwal, and A. M. Abu-Mahfouz, "Efficient two stage spectrum sensing combination for cognitive radio," in *In proceedings of the 27th International Symposium on Industrial Electronics (ISIE)*, pp. 1308–1313, June 2018.
9. X. Wang, H. Cheng, and M. Huang, "QoS multicast routing protocol oriented to cognitive network using competitive coevolutionary algorithm," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4513–4528, 2014.
10. J. J. Popoola and R. van Olst, "The performance evaluation of a spectrum sensing implementation using an automatic modulation classification detection method with a universal software radio peripheral," *Expert Systems with Applications*, vol. 40, no. 6, pp. 2165–2173, 2013.
11. D. Malafaia, J. Vieira, and A. Tomé, "Adaptive threshold spectrum sensing based on expectation maximization algorithm," *Physical Communication*, vol. 21, pp. 60–69, 2016.
12. J. Avila and K. Thenmozhi, "Adaptive double threshold with multiple energy detection technique in cognitive radio," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 10, no. 11, pp. 1336–1342, 2015.
13. A. J. Onumanyi, E. N. Onwuka, A. M. Aibinu, O. C. Ugweje, and M. J. E. Salami, "A modified Otsu's algorithm for improving the performance of the energy detector in cognitive radio," *AEU-International Journal of Electronics and Communications*, vol. 79, pp. 53–63, Sept. 2017.
14. E. U. Ogbodo, D. G. Dorrell, and A. M. Abu-Mahfouz, "Performance analysis of correlated multi-channels in cognitive radio sensor network based smart grid," in *in the 13th IEEE AFRICON conference*, (Cape Town, South Africa), pp. 1653–1658, Sept. 2017.
15. E. U. Ogbodo, D. Dorrell, and A. M. Abu-Mahfouz, "Cognitive Radio Based Sensor Network in Smart Grid: Architectures, Applications and Communication Technologies," *IEEE Access*, vol. 5, no. c, pp. 19084–19098, 2017.
16. J. Vartiainen and P. Henttu, "Estimation of Signal Detection Threshold by CME Algorithms," *IEEE 59th Vehicular Technology Conference (VTC 2004-Spring)*, vol. 3, no. 4, pp. 1654 – 1658 (Volume 3), 2004.
17. J. J. Lehtomäki, J. Vartiainen, M. Juntti, and H. Saarnisaari, "CFAR outlier detection with forward methods," *IEEE Transactions on Signal Processing*, vol. 55, pp. 4702–4706, sep 2007.
18. J. J. Lehtomäki, J. Vartiainen, and H. Saarnisaari, "Adaptive FCME-based threshold setting for energy detectors," No. 33, pp. 1 – 5, Oct. 2011.
19. H. Saamisaari and P. Henttu, "Impulse detection and rejection methods for radio systems," *In IEEE Military Communications Conference, 2003. MILCOM'03*, vol. 2, pp. 1126–1131, 2003.
20. J. Vartiainen, J. J. Lehtomäki, and H. Saarnisaari, "Double-Threshold Based Narrowband Signal Extraction," *2005 IEEE 61st Vehicular Technology Conference*, vol. 2, no. 1, pp. 5–9, 2005.
21. J. Vartiainen, S. Aromaa, H. Saarnisaari, and M. Juntti, "Performance evaluation of transform selective interference suppression," *Communications*, pp. 1–7, 2004.
22. J. Vartiainen, A. Sami, H. Saarnisaari, and M. Juntti, "Selection process of a transform selective interference suppression algorithm," in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004.*, pp. 220–223, IEEE, 2004.
23. H. Puska, H. Saarnisaari, and J. Iinatti, "Comparison of antenna array algorithms in DS/SS code acquisition with jamming," in *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 2005, IEEE, 2005.
24. J. J. Lehtomäki, M. Juntti, and H. Saarnisaari, "CFAR strategies for channelized radiometer," *IEEE Signal Processing Letters*, vol. 12, pp. 13–16, Jan. 2005.
25. J. Vartiainen, H. Saarnisaari, J. J. Lehtomäki, and M. Juntti, "A blind signal localization and SNR estimation method," *Proc. IEEE Mil. Commun. Conf. (MILCOM'06)*, pp. 1–7, 2006.
26. J. J. Lehtomäki, J. Vartiainen, M. Juntti, and H. Saarnisaari, "Spectrum sensing with forward methods," in *Proceedings - IEEE Military Communications Conference MILCOM*, IEEE, oct 2007.
27. B. Shen, C. Zhao, L. Huang, K. Kwak, and Z. Zhou, "Wideband Primary User Signal Identification Approaches for Cognitive {MB}-{OFDM} {UWB} Systems," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, IEEE, nov 2008.
28. J. J. Lehtomäki, S. Salmenkaita, J. Vartiainen, J. P. Mäkelä, R. Vuohtoniemi, and M. Juntti, "Measurement

studies of a spectrum sensing algorithm based on double thresholding," in *2009 2nd International Workshop on Cognitive Radio and Advanced Spectrum Management, CogART 2009*, pp. 69–73, IEEE, may 2009.
29. J. Vartiainen, J. J. Lehtomäki, H. Saarnisaari, and M. Juntti, "Analysis of the consecutive mean excision algorithms," *Journal of Electrical and Computer Engineering*, vol. 2010, pp. 1–13, 2010.
30. K. X. Jia and Z. S. He, "Narrowband signal localization based on enhanced LAD method," *Journal of Communications and Networks*, vol. 13, pp. 6–11, feb 2011.
31. J. J. Lehtomäki, R. Vuohtoniemi, and K. Umebayashi, "Duty Cycle and Channel Occupancy Rate Estimation with MED-FCME LAD ACC," in *Proceedings of the 7th International Conference on Cognitive Radio Oriented Wireless Networks*, vol. 248454, pp. 236–241, IEEE, 2012.
32. J. Vartiainen, "Always One/Zero Malicious User Detection in Cooperative Sensing Using the {FCME} Method," in *Proceedings of the 7th International Conference on Cognitive Radio Oriented Wireless Networks*, IEEE, 2012.
33. L. Mucchi, A. Carpini, T. D'Anna, M. H. Virk, R. Vuohtoniemi, M. Hämäläinen, and J. Iinatti, "Threshold setting for the evaluation of the aggregate interference in ISM band in hospital environments," in *International Symposium on Medical Information and Communication Technology, ISMICT*, vol. 2015-May, pp. 20–24, IEEE, mar 2015.
34. L. Schlain, G. González, F. Gregorio, and J. Cousseau, "Adaptive cyclostationary filtering for DGPS interference cancellation," in *2015 16th Workshop on Information Processing and Control, RPIC 2015*, IEEE, oct 2016.
35. H. Iwata, K. Umebayashi, S. Tiiro, Y. Suzuki, and J. J. Lehtomäki, "A study on Welch FFT segment size selection method for spectrum awareness," *2016 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2016*, no. 8, pp. 252–257, 2016.
36. J. J. Lehtomäki, R. Vuohtoniemi, K. Umebayashi, and J. P. Mäkelä, "Energy detection based estimation of channel occupancy rate with adaptive noise estimation," *IEICE transactions on communications*, vol. E95-B, no. 4, pp. 1076–1084, 2012.
37. R. Vuohtoniemi, J. J. Lehtomäki, and J. P. Mäkelä, "Adaptive threshold based frequency exclusion algorithm for broadband {PLC}," in *2016 International Symposium on Power Line Communications and its Applications ({ISPLC})*, IEEE, mar 2016.
38. N. Veček, M. Mernik, B. Filipič, and M. Črepinšek, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms," *Information Sciences*, vol. 372, pp. 446–469, 2016.
39. M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 11–18, Morgan Kaufmann Publishers Inc., 2002.
40. V. Nannen and A. E. Eiben, "Efficient relevance estimation and value calibration of evolutionary algorithm parameters," in *2007 IEEE Congress on Evolutionary Computation*, pp. 103–110, IEEE, 2007.
41. M. Črepinšek, S.-H. Liu, L. Mernik, and M. Mernik, "Is a comparison of results meaningful from the inexact replications of computational experiments?," *Soft Computing*, vol. 20, no. 1, pp. 223–235, 2016.
42. D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proc. of MENDEL 2002, 8th Int. Conf. on Soft Computing*, pp. 62–67, 2002.
43. N. Gul, I. M. Qureshi, A. Omar, A. Elahi, and S. Khan, "History based forward and feedback mechanism in cooperative spectrum sensing including malicious users in cognitive radio network," *PloS One*, vol. 12, no. 8, pp. 1–21, 2017.
44. N. Gul, I. M. Qureshi, A. Elahi, and I. Rasool, "Defense against malicious users in cooperative spectrum sensing using genetic algorithm," *International Journal of Antennas and Propagation*, vol. 2018, 2018.
45. N. Gul, I. M. Qureshi, S. Akbar, M. Kamran, and I. Rasool, "One-to-many relationship based kullback leibler divergence against malicious users in cooperative spectrum sensing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
46. A. J. Onumanyi, A. M. Abu-Mahfouz, and G. P. Hancke, "A comparative analysis of local and global adaptive threshold estimation techniques for energy detection in cognitive radio," *Physical Communication*, vol. 29, pp. 1–11, Apr. 2018.
47. X. S. Yang and S. Deb, "Cuckoo search via *Lėvy* flights," *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, pp. 210–214, 2009.
48. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, jan 1979.
49. T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.
50. D. Karaboga, "An idea based on honey bee swarm for numerical optimization," tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.
51. R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942–1948, Citeseer, 1995.
52. R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.
53. IEEE802.22, "Enabling broadband wireless access using cognitive radio technology and spectrum sharing in white spaces," *IEEE 802.22 Working Group on Wireless Regional Area Networks*, 2011.
54. A. J. Onumanyi, "Dataset for testing the performance of a cuckoo search optimization based forward consecutive mean excision model for threshold adaptation in cognitive radio," *Mendeley Data, v1*, vol. DOI: 10.17632/jxpnhc43xr.1, Oct. 2018.